enumext

ENUMERATE EXERCISE SHEETS

V1.0   2024-10-04[*]

©2024 by Pablo González[†]

Abstract

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

Contents

Motivation and acknowledgments

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

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

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

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

---

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 `10`pt, `11`pt and `12`pt.

1   Introduction

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

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

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

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

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

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

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

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

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

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

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

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

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

1. First type of questions

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

2. Second type of questions

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

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

* 3. Third type of questions

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

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

4. Question with image and label below:

A)   B)   C)

D)   E)

5. Question with image on left side:

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

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

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

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

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

1.1　Description and usage

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

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

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

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

The package is loaded in the usual way:

```
\usepackage{enumext}
```

1.2　The concept of left margin

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

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

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

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

Figure 2: Representation of horizontal lengths concept in enumext.

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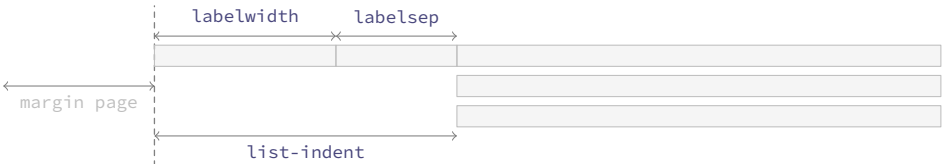

Figure 3: Default horizontal lengths `list-offset=0pt`, `list-indent=\labelwidth+\labelsep` in enumext.

1.3 User interface

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

1.3.1 Internal counters

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

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

1.3.2 Public dimension

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

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

1.3.3 Support for multicol

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

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

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

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

1.3.4 Support for minipage

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

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

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

1.3.5 The \label and \ref system

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

1.3.6 Support for \footnote

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

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

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

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

2 The environments provided

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

enumext
enumext*

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

2.1 The environment enumext

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

Example with `columns=2`

1. This text is in the first level.

    (a) This text is in the second level.

       i. This text is in the third level.

A. This text is in the fourth level.

X This text is in the first level.

* 2. This text is in the first level.

2.2 The environment enumext*

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

Some considerations to take into account for this environment:
- The environment cannot be nested within itself or in the environment `keyans*`, but it can be nested within `enumext` and vice versa.
- Each *"item 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.
- You cannot have any standard list environments like `itemize`, `enumerate`, `description`, `quote`, `quotation`, `verse`, `center`, `flushleft`, `flushright`, `verbatim`, `tabbing`, `trivlist`, `list` and all environments created with `\newtheorem`.

Example with `columns=2`

1. This text is in the first level.

X This text is in the first level.

2. This text is in the first level.

* 4. This text is in the first level.

2.3 The command \item*

\item*

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

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

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

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

2.3.1 Keys for \item*

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

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 enum**ext** package is implemented using `l3keys` so it must be taken into consideration that those keys marked as *"value forbidden"*, that is ⟨*key*⟩ is different from ⟨*key=*⟩.

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

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

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

5.1   Keys for `label` and `ref`

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

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

🌊 This key is intended to give the basic structure with which the ⟨*label*⟩ will be displayed, and the form in which it is used by standard *"label and ref"* and the *"internal 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 `label` key. Internally sets the value of `\labelwidth` for the current level. The default values are calculated by means of the *width* of a box by setting a *value* to the current counter using '`0`' for `\arabic*`, '`M`' for `\Alph*`, '`m`' for `\alph*`, '`VIII`' for `\Roman*` and '`viii`' for `\roman*`.

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

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

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

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

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

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

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

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

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

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

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

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

🌊 By default all the ⟨*keys*⟩ described above are executed inside `\makebox` in the `enumext*` and `keyans*` environments.

🌊 For compatibility with *tagged* PDF all ⟨*keys*⟩ described above are executed inside `\makebox` in the `enumext` and `keyans` environments, this means that the document output may *not look* the same when `\DocumentMetadata` is active. If you use the `wrap-label` or `wrap-label*` keys you can add conditional code using `\IfDocumentMetadataTF`.

`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 *"meta-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`.

🌀 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

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

Extra *horizontal indentation*, beyond `labelsep`, of the *"first line"* off each item. This value must be greater than or equal to `0pt` and is applied internally using `\hspace` without modifying the value of `\itemindent`.

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

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

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

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

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

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

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

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

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

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

5.3 Keys for add code

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

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

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

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

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

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

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

💣 Keep in mind that the code set in this key will affect the entire *"body"* of the environment and therefore the inner levels of the list and the `keyans` environment. It is recommended to set this key per level. In the `enumext*` and `keyans*` environments this key is executed within the `minipage` environment in which *"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 environment. The value must be a positive integer less than or equal to `10`.

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

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

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

5.6   Keys for `minipage`

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

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

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

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

5.6.1   The command `\miniright`

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

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

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

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

5.6.2 The key `mini-right`

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

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

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

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

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

6 The storage system

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

```
\begin{enumext}[save-ans={⟨store name⟩}]
  \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.

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: \※

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

6.1.2   Keys for `wrap` and `display`

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

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

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

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

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

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

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

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

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

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

6.1.3   Keys for debug and checking

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

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

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

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

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

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

6.2   The command `\anskey`

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

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

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

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

6.2.1 Keys for `\anskey`

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

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

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

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

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

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

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

`item-sym*` = {⟨*symbol*⟩}   default: *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.

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

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

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

6.4.1 The \item* in keyans and keyans*

\item* `\item*`
`\item*[⟨content⟩]`

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

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

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

Example

```
\begin{enumext}[save-ans=test,columns=2,show-ans=true]
  \item Text containing a question.
    \begin{keyans*}[nosep,columns=2]
      \item Choice
      \item* Correct choice
      \item Choice
      \item Choice
      \item Choice
    \end{keyans*}
  \item Text containing a question and image.
    \begin{keyans}[nosep,mini-env={0.4\linewidth}]
      \item Choice
      \item Choice
      \item Choice
      \item Choice
      \item*[⟨note⟩] Correct choice
      \miniright
      \includegraphics[scale=0.25]{example-image-a}
      Some text
    \end{keyans}
\end{enumext}
```

1. Text containing a question.

   A) Choice   * 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}*[⟨n° upper, n° lower⟩]\anspic{⟨drawing⟩}\anspic*[⟨content⟩]{⟨drawing or tabular⟩}

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

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

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

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

6.5.1 The command \anspic

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

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

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

Example

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

    \begin{keyanspic}[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.

    \begin{keyanspic}[label-pos=above, 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.

    \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.

A	B	A
A)	B)	C)

A	A
D)	* E)[note]

2. Question with images.

A)	B)	C)
A	B	A

D)	* E)[note]
A	A

3. Question with images.

A	B	A	A	A
A)	B)	C)	D)	* E)[note]

6.6 Printing stored content

6.6.1 The command \getkeyans

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

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

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

6.6.2 The command \foreachkeyans

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

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

Options for command

`sep = {`⟨*code*⟩`}` default: *{; }*

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 it will finish executing. The value must be a ⟨*positive integer*⟩.

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

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

`after = {`⟨*code*⟩`}` default: *empty*

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

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

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

6.6.3 The command \printkeyans

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

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

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

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

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

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

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

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

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

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

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

Example

```
\begin{enumext}[save-ans=sample,columns=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) L^AT_EX2e 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
   - C) value
   - B) correct
   - D) value

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

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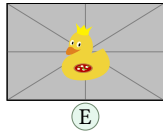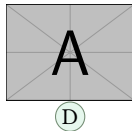   - (2) $\angle EDF = 45°$
   - Ⓐ value
   - Ⓑ value
   - Ⓒ value
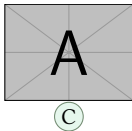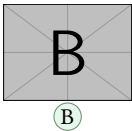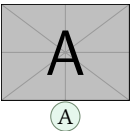   - Ⓓ value
   - Ⓔ value

4. Question with image and label below:

   Ⓐ   Ⓑ   Ⓒ   Ⓓ   Ⓔ

5. Question with image on left side:
   (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` and `hyperref` are required.

- The file `enumext-01.tex` contains the basic tests for the `enumext` and `enumet*` 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-01.tex` contains the tests for the `enumext` and `enumet*` 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 📄.

9 The way of non-enumerated lists

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

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

Fake `itemize` environment

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

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

* First level item
  ◇ Second level item
    ◦ Third level item
      ⋆ Fourth level item
* First level item

Fake `description` environment

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

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

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

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

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

Description indented by label

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

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

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

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

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

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

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

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

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

Description with multi-line labels

The `label` key does not accept *multiline material*, this is where the `wrap-label` 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 }
  {%
    \IfBooleanTF{#1}
      {\strut\smash{\parbox[t]{\labelwidth}{\raggedright{#2}}}}%
      {\strut\smash{\parbox[t]{\labelwidth}{\raggedleft{#2}}}}%
  }
```

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

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

Final notes

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

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

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

Why list environments?

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

Of course, seeing the final result of the experiment *"reinventing the wheel"* I am quite satisfied.

Why not random questions and other utilities

The *"random"* type questions I love and hate them at the same time, although they simplify a lot the work when creating a multiple choice test, but you lose the beauty of 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.

11 Change history

v1.0 2024-10-04   – 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}[2024-06-01]
```

Now declare the enumext package.

```
4  \ProvidesExplPackage
5    {enumext}
6    {2024-10-04}
7    {1.0}
8    {Enumerate exercise sheets}
```

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

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

13.4   Definition of variables

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

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

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

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

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

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

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

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

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

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

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

The initial values of these variables are set by the function __enumext_define_counters:Nn (§13.10) and then modified by the function __enumext_label_style:Nnn used by label key (§13.13).

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

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

\c__enumext_counter_style_tl
\l__enumext_ref_key_arg_tl
\l__enumext_ref_the_count_tl
\l__enumext_the_counter_X_tl
\l__enumext_renew_the_count_X_tl

Internal variables used by ref key (§13.13).

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

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

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

Internal variables used by resume, resume* and series keys (§13.24).

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

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

\l__enumext_current_widest_dim
\g__enumext_counter_styles_tl
\g__enumext_widest_label_tl
\l__enumext_label_width_by_box

The variable \l__enumext_current_widest_dim stores the current label width, the variable \g__enumext_counter_styles_tl stores the default ⟨*label style*⟩ and the variable \g__enumext_widest_label_tl the label width. These variables are used by widest (§13.14) and label (§13.12) keys.

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

(End of definition for \l__enumext_current_widest_dim and others.)

\l__enumext_leftmargin_tmp_X_bool
\l__enumext_leftmargin_tmp_X_dim
\l__enumext_leftmargin_X_dim
\l__enumext_itemindent_X_dim

The boolean variable \l__enumext_leftmargin_tmp_X_bool and the dimensional variable \l__enumext_-leftmargin_tmp_X_dim are used by the list-indent key (§13.17). The variables \l__enumext_-leftmargin_X_dim and \l__enumext_itemindent_X_dim are used and set by the function __enumext_-calc_hspace:NNNNNNNNNNNN (§13.37.1).

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

(End of definition for \l__enumext_leftmargin_tmp_X_bool and others.)

\l__enumext_multicols_above_X_skip
\l__enumext_multicols_below_X_skip
\g__enumext_multicols_right_X_skip
\l__enumext_align_label_pos_X_str

Internal variables used by columns key (§13.21) and align key (§13.12).

```
79  \cs_set_protected:Npn \__enumext_tmp:n #1
80    {
81      \skip_new:c  { l__enumext_multicols_above_#1_skip }
82      \skip_new:c  { l__enumext_multicols_below_#1_skip }
83      \skip_new:c  { g__enumext_multicols_right_#1_skip }
84      \str_new:c   { l__enumext_align_label_pos_#1_str  }
85    }
86  \clist_map_inline:nn { i, ii, iii, iv, v } { \__enumext_tmp:n {#1} }
```

(End of definition for \l__enumext_multicols_above_X_skip and others.)

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

Internal variables used by \miniright command (§13.22.4) and the keys mini-right, mini-right*, mini-env and mini-sep (§13.20, §13.22).

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

(End of definition for \g__enumext_minipage_stat_int and others.)

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

The bool vars \l__enumext_wrap_label_X_bool and \l__enumext_wrap_label_opt_X_bool are used by wrap-label and wrap-label* keys (§13.12), the integer \l__enumext_start_X_int are used by the start and start* keys (§13.14), the token list \l__enumext_fake_item_indent_X_tl is used by itemindent key (§13.17.1), the variables \l__enumext_label_fill_left_X_tl and \l__enumext_-label_fill_left_X_tl are used by the align key (§13.12). The boolean vars \l__enumext_vspace_-a_star_X_bool, \l__enumext_vspace_b_star_X_bool are used by above, above*, below and below* keys (§13.19).

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

(End of definition for \l__enumext_wrap_label_X_bool and others.)

The variable `\l__enumext_store_active_bool` setting by save-ans key (§13.25.1) activates all the mechanism related to \anskey, anskey*, keyans, keyans* and keyanspic environments.

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

The variable `\l__enumext_store_anskey_arg_tl` save the *argument* of \anskey (§13.29) and the variables `\l__enumext_store_anskey_env_tl` and `\l__enumext_store_anskey_opt_tl` save the ⟨*body*⟩ and the ⟨*keys*⟩ of the environment anskey* (§13.30).

The variables `\l__enumext_store_current_label_tl` and `\l__enumext_store_current_opt_arg_-tl` save the *current label* and *optional argument* of \item* (§13.36) and \anspic* (§13.41.2) for the keyans, keyans* and keyanspic environments.

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

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

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

Internal variables used by the command \setenumext (§13.47).

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

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

Internal variables used by the \printkeyans command (§13.46) and \foreachkeyans command (§13.49).

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

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

Internal variables used by command \printkeyans (§13.46), show-pos key (§13.26), item-sym* key (§13.34), save-key key (§13.26.2) and *"storing structure"*.

```
130 \tl_new:N   \l__enumext_print_keyans_starred_tl
131 \bool_new:N \l__enumext_print_keyans_star_bool
132 \str_new:N  \l__enumext_mark_position_str
133 \tl_new:N   \g__enumext_item_symbol_aux_tl
134 \cs_set_protected:Npn \__enumext_tmp:n #1
135   {
136     \tl_new:c   { l__enumext_print_keyans_#1_tl        }
137     \tl_new:c   { l__enumext_store_save_key_#1_tl       }
138     \bool_new:c { l__enumext_store_save_key_#1_bool     }
139     \bool_new:c { l__enumext_store_upper_level_#1_bool  }
140   }
141 \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 (§13.41.1).

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

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

\l__enumext_check_answers_bool
\g__enumext_check_ans_key_bool
\l__enumext_check_start_line_env_tl
\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 (§13.25.3) used by the check-ans and no-store keys and check for starred commands \item* in keyans and keyans* environments and \anspic* in keyanspic environment.

```
153 \bool_new:N \l__enumext_check_answers_bool
154 \bool_new:N \g__enumext_check_ans_key_bool
155 \tl_new:N   \l__enumext_check_start_line_env_tl
156 \int_new:N  \g__enumext_check_starred_cmd_int
157 \int_new:N  \g__enumext_item_anskey_int
158 \int_new:N  \g__enumext_item_number_int
159 \bool_new:N \l__enumext_item_number_bool
160 \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 (§13.8). The boolean variable \l__enumext_footnotes_key_bool determine if hyperref is load with key hyperfootnotes=true.

```
161 \bool_new:N \l__enumext_hyperref_bool
162 \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 (§13.26). The variables \l__enumext_label_copy_X_tl correspond to temporary copies of the ⟨*labels*⟩ defined by level on which operations will be performed.

The variables \l__enumext_newlabel_arg_one_tl and \l__enumext_newlabel_arg_two_tl will be used to form the arguments passed to the function __enumext_newlabel:nn (§13.8) and the variable \l__enumext_write_aux_file_tl will be in charge of executing the writing code in the .aux file.

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

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

\g__enumext_footnote_int
\g__enumext_footnote_arg_seq
\g__enumext_footnote_int_seq

Internal variables used for redefinition of \footnote (§13.42.4).

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

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

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

Internal variables used by enumext* and keyans* environments.

```
174 \cs_set_protected:Npn \__enumext_tmp:n #1
175   {
176     \bool_new:c { l__enumext_item_starred_#1_bool    }
177     \int_new:c  { l__enumext_item_column_pos_#1_int  }
178     \int_new:c  { g__enumext_item_count_all_#1_int   }
179     \int_new:c  { l__enumext_joined_item_#1_int      }
180     \int_new:c  { l__enumext_joined_item_aux_#1_int  }
181     \int_new:c  { l__enumext_tmpa_#1_int             }
182     \dim_new:c  { l__enumext_tmpa_#1_dim             }
183     \box_new:c  { l__enumext_item_text_#1_box        }
184     \dim_new:c  { l__enumext_joined_width_#1_dim     }
185     \dim_new:c  { l__enumext_item_width_#1_dim       }
186     \tl_new:c   { g__enumext_item_symbol_aux_#1_tl   }
187     \str_new:c  { l__enumext_align_label_#1_str      }
188     \bool_new:c { g__enumext_minipage_active_#1_bool }
189     \box_new:c  { l__enumext_miniright_code_#1_box   }
190     \bool_new:c { g__enumext_minipage_center_#1_bool }
191     \dim_new:c  { g__enumext_minipage_right_#1_dim   }
192     \skip_new:c { g__enumext_minipage_right_#1_skip  }
193   }
194 \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`.

```
195 \clist_const:Nn \c__enumext_all_envs_clist
196   {
197     {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv},
198     {keyans}{v}, {enumext*}{vii}, {keyans*}{viii}
199   }
```

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

13.5   Some utility functions

`\keys_precompile:neN` Non-standard kernel variants used by the `\printkeyans` command (§13.46) and `\foreachkeyans` command
`\seq_use:NV` (§13.49).

```
200 \cs_generate_variant:Nn \keys_precompile:nnN { neN }
201 \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.

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

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

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

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

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

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

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

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

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

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

(*End of definition for* `\__enumext_if_is_int:nT`, `\__enumext_if_is_int:nF`, *and* `\__enumext_if_is_int:nTF`.)

`\__enumext_regex_counter_style:` The internal function `\__enumext_regex_counter_style:` replace the '*' with the actual counter of the running level and is used by the `ref` key. It loops through the defined counter styles in `\c__enumext_-counter_style_tl` and replace '*' by real command, for example, looking for `\arabic*` and replacing that by `\arabic{`⟨*counter*⟩`}` defined on the current level.

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

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

__enumext_show_length:nnn   Internal function used by `show-length` key to show *"all lengths"* calculated and use in `enumext`, `enumext*`, `keyans` and `keyans*` environments.

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

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

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

```
238  \cs_new_protected:Nn \__enumext_unskip_unkern:
239    {
240      \int_case:nnT { \lastnodetype }
241        {
242          { 11 }
243            {
244              % \typeout{SKIP} \typeout{\the\lastskip}
245              \unskip
246            }
247          { 12 }
248            {
249              % \typeout{KERN} \typeout{\the\lastkern}
250              \unkern
251            }
252        }
253    }
```

(*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 nested in each other and save the environment name in \l__enumext_envir_name_tl.

```
254  \cs_new_protected:Nn \__enumext_is_not_nested:
255    {
256      \str_case:en { \@currenvir }
257        {
258          {enumext}
259            {
260              \tl_set:Nn \l__enumext_envir_name_tl { enumext }
261              \bool_lazy_and:nnT
262                { \bool_not_p:n { \g__enumext_standar_bool } }
263                { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
264                {
265                  \bool_gset_true:N \g__enumext_standar_bool
266                }
267            }
268          {enumext*}
269            {
270              \tl_set:Nn \l__enumext_envir_name_tl { enumext* }
271              \bool_lazy_and:nnT
272                { \bool_not_p:n { \g__enumext_starred_bool } }
273                { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
274                {
275                  \bool_gset_true:N \g__enumext_starred_bool
276                }
277            }
278        }
279    }
```

The function __enumext_is_on_first_level: will set the variables \l__enumext_standar_first_-bool (§13.25.1), \l__enumext_starred_first_bool (§13.25.1) and \l__enumext_anskey_env_bool (§13.30) to *"true"* only if the environment is not nested and we are in the *"first level"* of it . We will also save the *start line number* of each environment in the variable \g__enumext_start_line_tl and the *name* of each environment in the variable \g__enumext_envir_name_tl to use in messages related to the `check-ans` key and `.log` file.

```
280  \cs_new_protected:Nn \__enumext_is_on_first_level:
281    {
282      \bool_lazy_all:nT
283        {
284          { \bool_if_p:N \g__enumext_standar_bool }
285          { \int_compare_p:nNn { \l__enumext_level_int } = { 1 } }
286          { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
287        }
288        {
289          \bool_set_true:N \l__enumext_standar_first_bool
290          \bool_set_true:N \l__enumext_anskey_env_bool
291          \tl_gset:Nn \g__enumext_envir_name_tl { enumext }
292          \tl_gset:Ne \g__enumext_start_line_tl
293            {
294              on ~ line ~ \exp_not:V \inputlineno
295            }
296        }
297      \bool_lazy_all:nT
298        {
299          { \bool_if_p:N \g__enumext_starred_bool }
300          { \int_compare_p:nNn { \l__enumext_level_h_int } = { 1 } }
301          { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
302        }
303        {
304          \bool_set_true:N \l__enumext_starred_first_bool
305          \bool_set_true:N \l__enumext_anskey_env_bool
306          \tl_gset:Nn \g__enumext_envir_name_tl { enumext* }
307          \tl_gset:Ne \g__enumext_start_line_tl
308            {
309              on ~ line ~ \exp_not:V \inputlineno
310            }
311        }
312    }
```

(*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.

```
313  \cs_new_protected:Nn \__enumext_keyans_name_and_start:
314    {
315      \str_case:en { \@currenvir }
316        {
317          {keyans}
318            {
319              \tl_set:Nn \l__enumext_envir_name_tl { keyans }
320              \tl_set:Ne \l__enumext_check_start_line_env_tl
321                {
322                  in ~ 'keyans' ~ start ~ on ~ line ~ \exp_not:V \inputlineno
323                }
324            }
325          {keyans*}
326            {
327              \tl_set:Nn \l__enumext_envir_name_tl { keyans* }
328              \tl_set:Ne \l__enumext_check_start_line_env_tl
329                {
330                  in ~ 'keyans*' ~ start ~ on ~ line ~ \exp_not:V \inputlineno
331                }
332            }
333          {keyanspic}
334            {
335              \tl_set:Nn \l__enumext_envir_name_tl { keyanspic }
336              \tl_set:Ne \l__enumext_check_start_line_env_tl
337                {
338                  in ~ 'keyanspic' ~ start ~ on ~ line ~ \exp_not:V \inputlineno
339                }
340            }
341        }
342    }
```

(*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.

```
343 \cs_new_protected:Nn \__enumext_reset_global_vars:
344   {
345     \__enumext_reset_global_int:
346     \__enumext_reset_global_bool:
347     \__enumext_reset_global_tl:
348   }
349 \cs_new_protected:Nn \__enumext_reset_global_int:
350   {
351     \int_gzero:N \g__enumext_item_number_int
352     \int_gzero:N \g__enumext_item_anskey_int
353     \int_gzero:N \g__enumext_item_answer_diff_int
354   }
355 \cs_new_protected:Nn \__enumext_reset_global_bool:
356   {
357     \bool_gset_false:N \g__enumext_check_ans_key_bool
358     \bool_gset_false:N \g__enumext_standar_bool
359     \bool_gset_false:N \g__enumext_starred_bool
360   }
361 \cs_new_protected:Nn \__enumext_reset_global_tl:
362   {
363     \tl_gclear:N \g__enumext_store_name_tl
364     \tl_gclear:N \g__enumext_start_line_tl
365     \tl_gclear:N \g__enumext_envir_name_tl
366   }
```

(*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.

```
367 \cs_new_protected:Nn \__enumext_log_global_vars:
368   {
369     \msg_log:nneeee { enumext } { prop-seq-int-hook }
370       { \g__enumext_store_name_tl }
371       { \prop_count:c { g__enumext_ \g__enumext_store_name_tl _prop } }
372       { \seq_count:c { g__enumext_ \g__enumext_store_name_tl _seq } }
373       { \int_use:c { g__enumext_resume_ \g__enumext_store_name_tl _int } }
374   }
```

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.

```
375 \cs_new_protected:Nn \__enumext_log_answer_vars:
376   {
377     \msg_log:nneee { enumext } { item-answer-hook }
378       { \int_use:N \g__enumext_item_number_int }
379       { \int_use:N \g__enumext_item_anskey_int }
380       { \int_eval:n { \g__enumext_item_number_int - \g__enumext_item_anskey_int} }
381   }
```

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

13.6 Copying `list` and `minipage` environments

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

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

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

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

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

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

__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, the function __enumext_item_std:w is a copy of the \item command.

```
382 \__enumext_at_begin_document:n
383   {
384     \cs_new_eq:NN   \__enumext_start_list:nn \list
385     \cs_new_eq:NN   \__enumext_stop_list: \endlist
386     \NewCommandCopy \__enumext_item_std:w \item
387   }
```

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

```
388 \__enumext_at_begin_document:n
389   {
390     \cs_new_eq:NN \__enumext_minipage:w \minipage
391     \cs_new_eq:NN \__enumext_endminipage: \endminipage
392   }
```

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

13.7   The internal minipage environment

__enumext_internal_mini_page:
__enumext_mini_env*

The function __enumext_internal_mini_page: creates a internal __enumext_mini_page environment (*custom version* of minipage) setting the \if@minipage switch to *"false"* to allow spaces at the *"above"* of the environment, plus we will add \skip_vertical:N \c_zero_skip to maintain alignment on *"top"* in the first part and \skip_vertical:N \c_zero_skip in the second part to allow spaces *"below"*. This environment will be used internally by the mini-env key, it is not documented in the user interface and is for internal use only. This function is passed to the function __enumext_safe_exec: in the enumext environment definition (§13.38) and __enumext_safe_exec_vii: in the enumext* environment definition (§13.43)

```
393 \cs_new_protected:Nn \__enumext_internal_mini_page:
394   {
395     \int_compare:nNnT { \l__enumext_level_int } = { 0 }
396       {
397         \DeclareDocumentEnvironment{__enumext_mini_page}{ m }
398           {
399             \__enumext_minipage:w [ t ] { ##1 }
400               \legacy_if_gset_false:n { @minipage }
401               \skip_vertical:N \c_zero_skip
402           }
403           {
404               \skip_vertical:N \c_zero_skip
405             \__enumext_endminipage:
406           }
407       }
408   }
```

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

13.8   Compatibility with hyperref and footnotehyper

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

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

__enumext_after_hyperref:
__enumext_hypertarget:nn
__enumext_phantomsection:

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

```
411 \cs_new_protected:Nn \__enumext_after_hyperref:
412   {
413     \IfPackageLoadedTF { hyperref }
414       {
415         \msg_info:nnn { enumext } { package-load } { hyperref }
416         \bool_set_true:N \l__enumext_hyperref_bool
417         \IfHyperBoolean{hyperfootnotes}
418           {
419             % \typeout{hyperfootnotes=true}
420             \bool_set_true:N \l__enumext_footnotes_key_bool
421           }
422           {
```

```
423        % \typeout{hyperfootnotes=false}
424      }
425    }
426    {  }
```

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.

```
427    \bool_if:NT \l__enumext_footnotes_key_bool
428      {
429        \IfPackageLoadedTF { footnotehyper }
430          {
431            \msg_info:nnn { enumext } { package-load } { footnotehyper }
432          }
433          {
434            % \typeout{No ~ footnotehyper ~ load}
435            % \typeout{Load ~ and  ~ use  ~ \string\makesavenoteenv{enumext*}}
436            \bool_set_false:N \l__enumext_footnotes_key_bool
437          }
438      }
```

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.

```
439    \bool_if:NTF \l__enumext_hyperref_bool
440      {
441        \cs_new_eq:NN \__enumext_hypertarget:nn \hypertarget
442        \cs_new_eq:NN \__enumext_phantomsection: \phantomsection
443      }
444      {
445        \cs_new_eq:NN \__enumext_hypertarget:nn \use_none:nn
446        \cs_new_eq:NN \__enumext_phantomsection: \prg_do_nothing:
447      }
448    }
```

(*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.

```
449  \cs_new_protected:Npn \__enumext_newlabel:nn #1 #2
450    {
451      \protected@write \@auxout { }
452        {
453          \token_to_str:N \newlabel {#1}
454            {
455              {#2}
456              \bool_if:NT \l__enumext_hyperref_bool
457                { { \thepage } {#2} {#1} }
458              { }
459            }
460        }
461      \__enumext_hypertarget:nn {#1} { }
462      \__enumext_phantomsection:
463    }
```

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

13.9   Definition of public dimension

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

```
464  \dim_zero_new:N \itemwidth
```

13.10 Definition of counters

__enumext_define_counters:Nn
__enumext_define_counters:cn

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

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

#2 : The counter's name.

```
465 \cs_new_protected:Npn \__enumext_define_counters:Nn #1 #2
466   {
467     \cs_if_exist:cTF { c@ #2 }
468       { \msg_fatal:nnn { enumext } { counters }{ #2 } }
469       {
470         \tl_set:Nn #1 { #2 }
471         \newcounter { #2 }
472       }
473   }
```

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

enumXi
enumXii
enumXiii
enumXiv
enumXv
enumXvi
enumXvii
enumXviii

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

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

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

13.11 Definition of labels

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

__enumext_register_counter_style:Nn

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

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

```
492 \cs_new_protected:Npn \__enumext_label_width_by_box:Nn #1 #2
493   {
494     \hbox_set:Nn \l__enumext_label_width_by_box {#2}
495     \dim_set:Nn #1 { \box_wd:N \l__enumext_label_width_by_box }
496   }
497 \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.

```
498 \cs_new_protected:Npn \__enumext_label_style:Nnn #1 #2 #3
```

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

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

13.12   Setting keys associated with label

When *tagged* PDF is active \makelabel is redefined using \makebox to work correctly (§13.33). 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 environment.

```
514  \keys_define:nn { enumext /  level-1 }
515    {
516      mode-box .bool_set:N = \l__enumext_mode_box_bool,
517      mode-box .initial:n  = false,
518      mode-box .value_forbidden:n = true,
519    }
```

(*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.

```
520  \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
521    {
522      \keys_define:nn { enumext / #1 }
523        {
524          font        .tl_set:c   = { l__enumext_label_font_style_#2_tl },
525          font        .value_required:n = true,
526          labelsep    .dim_set:c  = { l__enumext_labelsep_#2_dim },
527          labelsep    .initial:n  = {0.3333em},
528          labelsep    .value_required:n = true,
529          labelwidth  .dim_set:c  = { l__enumext_labelwidth_#2_dim },
530          labelwidth  .value_required:n = true,
531          wrap-label  .cs_set_protected:cp = { __enumext_wrapper_label_#2:n } ##1,
532          wrap-label  .initial:n  = {##1},
533          wrap-label  .value_required:n = true,
534          wrap-label* .code:n = {
535                                    \bool_set_true:c { l__enumext_wrap_label_opt_#2_bool }
536                                    \keys_set:nn { enumext / #1 } { wrap-label = {##1} }
537                                  },
538          wrap-label* .value_required:n = true,
539        }
540    }
541  \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }
```

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

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

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

```
542  \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
543    {
544      \keys_define:nn { enumext / #1 }
545        {
546          align .choice:,
```

```
547            align / left     .code:n =
548                                {
549                                  \tl_clear:c { l__enumext_label_fill_left_#2_tl  }
550                                  \tl_set:cn  { l__enumext_label_fill_right_#2_tl } { \hfill }
551                                  \str_set:cn { l__enumext_align_label_pos_#2_str } { l }
552                                },
553            align / right    .code:n =
554                                {
555                                  \tl_set:cn  { l__enumext_label_fill_left_#2_tl  } { \hfill }
556                                  \tl_clear:c { l__enumext_label_fill_right_#2_tl }
557                                  \str_set:cn { l__enumext_align_label_pos_#2_str } { r }
558                                },
559            align / center   .code:n =
560                                {
561                                  \tl_set:cn { l__enumext_label_fill_left_#2_tl  } { \hfill }
562                                  \tl_set:cn { l__enumext_label_fill_right_#2_tl } { \hfill }
563                                  \str_set:cn { l__enumext_align_label_pos_#2_str } { c }
564                                },
565            align / unknown .code:n =
566                               \msg_error:nneee { enumext } { unknown-choice }
567                                 { align } { left, ~ right, ~  center } { \exp_not:n {##1} },
568            align .initial:n  = left,
569            align .value_required:n  = true,
570          }
571      }
572  \clist_map_inline:nn
573      {
574        {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {keyans}{v}
575      }
576      { \__enumext_tmp:nn #1 }

577  \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
578      {
579        \keys_define:nn { enumext / #1 }
580          {
581            align .choice:,
582            align / left     .code:n = \str_set:cn { l__enumext_align_label_#2_str } { l },
583            align / right    .code:n = \str_set:cn { l__enumext_align_label_#2_str } { r },
584            align / center   .code:n = \str_set:cn { l__enumext_align_label_#2_str } { c },
585            align / unknown .code:n =
586                               \msg_error:nneee { enumext } { unknown-choice }
587                                 { align } { left, ~ right, ~  center } { \exp_not:n {##1} },
588            align .initial:n = left,
589            align .value_required:n = true,
590          }
591      }
592  \clist_map_inline:nn { {enumext*}{vii}, {keyans*}{viii} } { \__enumext_tmp:nn #1 }
```

(*End of definition for* align.)

13.13   Setting label and ref keys

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

13.13.1   Define and set label and ref keys for enumext environment

label
ref
\l__enumext_label_i_tl
\l__enumext_label_ii_tl
\l__enumext_label_iii_tl
\l__enumext_label_iv_tl

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

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

```
606       ref    .value_required:n = true,
607     }
608   }
609 \__enumext_tmp:nnn { level-1 } {   i } { \arabic*.}
610 \__enumext_tmp:nnn { level-2 } {  ii } { (\alph*) }
611 \__enumext_tmp:nnn { level-3 } { iii } { \roman*. }
612 \__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.

```
613 \cs_new_protected:Npn \__enumext_standar_ref:n #1
614   {
615     \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
616     \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
617       {
618         \msg_error:nnn { enumext } { key-ref-empty } { enumext }
619       }
620       {
621         \tl_set_eq:Nc
622           \l__enumext_ref_the_count_tl { l__enumext_counter_ \__enumext_level: _tl }
623         \__enumext_regex_counter_style:
624         \tl_set_eq:Nc
625           \l__enumext_ref_the_count_tl { l__enumext_the_counter_ \__enumext_level: _tl }
626         \tl_put_right:ce { l__enumext_renew_the_count_ \__enumext_level: _tl }
627           {
628             \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl } { \exp_not:V \l__
629           }
630       }
631   }
```

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

```
632 \cs_new_protected:Nn \__enumext_standar_ref:
633   {
634     \tl_if_empty:cF { l__enumext_renew_the_count_ \__enumext_level: _tl }
635       {
636         \tl_use:c { l__enumext_renew_the_count_ \__enumext_level: _tl }
637       }
638   }
```

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

13.13.2 Define and set label and ref keys for enumext* and keyans* environments

label
ref
\l__enumext_label_vii_tl
\l__enumext_label_viii_tl

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

```
639 \cs_set_protected:Npn \__enumext_tmp:nnn #1 #2 #3
640   {
641     \keys_define:nn { enumext / #1 }
642       {
643         label .code:n    = {
644                           \__enumext_label_style:cvn { l__enumext_label_#2_tl }
645                           { l__enumext_counter_#2_tl } {##1}
646                           \dim_set_eq:cN  { l__enumext_labelwidth_#2_dim }
647                           \l__enumext_current_widest_dim
648                         },
649         label .initial:n = #3,
650         label .value_required:n = true,
651         ref   .code:n    = \__enumext_starred_ref:n {##1},
652         ref   .value_required:n = true,
653       }
654   }
655 \__enumext_tmp:nnn { enumext* } {  vii } { \arabic*.}
656 \__enumext_tmp:nnn { keyans*  } { viii } { \Alph*) }
```

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

`\__enumext_starred_ref:n`
`\__enumext_starred_ref:`

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

```
657 \cs_new_protected:Npn \__enumext_starred_ref:n #1
658   {
659     \tl_set:Nn \l_enumext_ref_key_arg_tl {#1}
660     \int_compare:nNnT { \l__enumext_level_h_int } = { 1 }
661       {
662         \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
663           {
664             \msg_error:nnn { enumext } { key-ref-empty } { enumext* }
665           }
666           {
667             \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_counter_vii_tl
668             \__enumext_regex_counter_style:
669             \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_the_counter_vii_tl
670             \tl_put_right:Ne \l__enumext_renew_the_count_vii_tl
671               {
672                 \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl } { \exp_not:V
673               }
674           }
675       }
676     \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
677       {
678         \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
679           {
680             \msg_error:nnn { enumext } { key-ref-empty } { keyans* }
681           }
682           {
683             \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_counter_viii_tl
684             \__enumext_regex_counter_style:
685             \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_the_counter_viii_tl
686             \tl_put_right:Ne \l__enumext_renew_the_count_viii_tl
687               {
688                 \exp_not:N \renewcommand  { \exp_not:V \l__enumext_ref_the_count_tl } { \exp_not:V
689               }
690           }
691       }
692   }
```

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.

```
693 \cs_new_protected:Nn \__enumext_starred_ref:
694   {
695     \int_compare:nNnT { \l__enumext_level_h_int } = { 1 }
696       {
697         \tl_if_empty:NF \l__enumext_renew_the_count_vii_tl
698           {
699             \tl_use:N \l__enumext_renew_the_count_vii_tl
700           }
701       }
702     \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
703       {
704         \tl_if_empty:NF \l__enumext_renew_the_count_viii_tl
705           {
706             \tl_use:N \l__enumext_renew_the_count_viii_tl
707           }
708       }
709   }
```

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

13.13.3 Define and set label and ref keys for keyans and keyanspic environments

label
ref
`\l__enumext_label_v_tl`
`\l__enumext_label_vi_tl`

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

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

```
718                            { l__enumext_counter_vi_tl } {#1}
719                        \dim_set_eq:cN  { l__enumext_labelwidth_v_dim }
720                            \l__enumext_current_widest_dim
721                    },
722    label .initial:n = \Alph*),
723    label .value_required:n = true,
724    ref    .code:n     = \__enumext_keyans_ref:n {#1},
725    ref    .value_required:n = true,
726  }
```

(*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.

```
727 \cs_new_protected:Npn \__enumext_keyans_ref:n #1
728   {
729     \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
730     \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
731       {
732         \msg_error:nnn { enumext } { key-ref-empty } { keyans }
733       }
734       {
735         \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_counter_v_tl
736         \__enumext_regex_counter_style:
737         \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_the_counter_v_tl
738         \tl_put_right:Ne \l__enumext_renew_the_count_v_tl
739           {
740             \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl } { \exp_not:V \l_
741           }
742       }
743   }
```

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

```
744 \cs_new_protected:Nn \__enumext_keyans_ref:
745   {
746     \tl_if_empty:NF \l__enumext_renew_the_count_v_tl
747       {
748         \tl_use:N \l__enumext_renew_the_count_v_tl
749       }
750   }
```

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

13.14   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.

```
751 \cs_new_protected:Npn \__enumext_start_from:NNn #1 #2 #3
752   {
753     \__enumext_if_is_int:nTF { #3 }
754       {
755         \int_set:Nn #2 {#3}
756       }
757       {
758         \regex_match:nVT { \c{Alph} | \c{alph} } {#1}
759           { \int_set:Nn #2 { \int_from_alph:n {#3} } }
760         \regex_match:nVT { \c{Roman} | \c{roman} } {#1}
761           { \int_set:Nn #2  { \int_from_roman:n {#3} } }
762       }
763   }
764 \cs_generate_variant:Nn \__enumext_start_from:NNn { ccn, cce }
```

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

__enumext_widest_from:nNNn
__enumext_widest_from:nccn

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

#1:   The counter associated with the environment level
#2: \l__enumext_label_X_tl

#3 : `\l__enumext_labelwidth_X_dim`

#4 : ⟨*integer or string*⟩

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

```
765 \cs_new_protected:Npn \__enumext_widest_from:nNNn #1 #2 #3 #4
766   {
767     \__enumext_if_is_int:nTF {#4}
768       {
769         \setcounter{enumX#1} { #4 }
770       }
771       {
772         \regex_match:nVT { \c{Alph} | \c{alph} } {#2}
773           { \setcounter{enumX#1} { \int_from_alph:n {#4} } }
774         \regex_match:nVT { \c{Roman} | \c{roman} } {#2}
775           { \setcounter{enumX#1} { \int_from_roman:n {#4} } }
776       }
777     \__enumext_label_width_by_box:cv
778       { l__enumext_labelwidth_#1_dim } { l__enumext_label_#1_tl }
779   }
780 \cs_generate_variant:Nn \__enumext_widest_from:nNNn { nccn }
```

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

start
start*
widest

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

```
781 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
782   {
783     \keys_define:nn { enumext / #1 }
784       {
785         start* .code:n    = {
786                             \__enumext_start_from:ccn
787                               { l__enumext_label_#2_tl }
788                               { l__enumext_start_#2_int } {##1}
789                           },
790         start* .value_required:n = true,
791         start  .code:n    = {
792                             \__enumext_start_from:cce
793                               { l__enumext_label_#2_tl }
794                               { l__enumext_start_#2_int } { \int_eval:n {##1} }
795                           },
796         start  .initial:n = 1,
797         start  .value_required:n = true,
798         widest .code:n    = {
799                             \__enumext_widest_from:nccn {#2}
800                               { l__enumext_label_#2_tl }
801                               { l__enumext_labelwidth_#2_dim } {##1}
802                           },
803         widest .value_required:n = true,
804       }
805   }
806 \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }
```

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

13.15 Setting keys for vertical spaces

topsep
partopsep
parsep
noitemsep
nosep

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

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

```
817          parsep    .skip_set:c = { l__enumext_parsep_#2_skip },
818          parsep    .initial:n  = {#5},
819          parsep    .value_required:n = true,
820          itemsep   .skip_set:c = { l__enumext_itemsep_#2_skip },
821          itemsep   .initial:n  = {#6},
822          itemsep   .value_required:n = true,
823          noitemsep .meta:n      = { itemsep = 0pt, parsep = 0pt },
824          noitemsep .value_forbidden:n = true,
825          nosep     .meta:n      = {
826                                    itemsep = 0pt, parsep= 0pt,
827                                    topsep = 0pt, partopsep = 0pt,
828                                 },
829          nosep     .value_forbidden:n = true,
830        }
831    }
```

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

```
832 \__enumext_tmp:nnnnnn { level-1 } { i } { 8.0pt plus 2.0pt minus 4.0pt }
833    { 2.0pt plus 1.0pt minus 1.0pt } { 4.0pt plus 2.0pt minus 1.0pt }
834    { 4.0pt plus 2.0pt minus 1.0pt }
835 \__enumext_tmp:nnnnnn { level-2 } { ii } { 4.0pt plus 2.0pt minus 1.0pt }
836    { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
837    { 2.0pt plus 1.0pt minus 1.0pt }
838 \__enumext_tmp:nnnnnn { level-3 } { iii } { 2.0pt plus 1.0pt minus 1.0pt }
839    { 1.0pt minus 1.0pt }{ 0pt }{ 2.0pt plus 1.0pt minus 1.0pt }
840 \__enumext_tmp:nnnnnn { level-4 } { iv } { 2.0pt plus 1.0pt minus 1.0pt }
841    { 1.0pt minus 1.0pt }{ 0pt }{ 2.0pt plus 1.0pt minus 1.0pt }
842 \__enumext_tmp:nnnnnn { keyans  } { v }{ 4.0pt plus 2.0pt minus 1.0pt }
843    { 2.0pt plus 1.0pt minus 1.0pt }{ 2.0pt plus 1.0pt minus 1.0pt }
844    { 2.0pt plus 1.0pt minus 1.0pt }
845 \__enumext_tmp:nnnnnn { enumext* } { vii } { 8.0pt plus 2.0pt minus 4.0pt }
846    { 2.0pt plus 1.0pt minus 1.0pt } { 4.0pt plus 2.0pt minus 1.0pt }
847    { 4.0pt plus 2.0pt minus 1.0pt }
848 \__enumext_tmp:nnnnnn { keyans* } { viii } { 4.0pt plus 2.0pt minus 1.0pt }
849    { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
850    { 2.0pt plus 1.0pt minus 1.0pt }
```

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

13.16 Setting `base-fix` key

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

```
851 \keys_define:nn { enumext /  level-1 }
852    {
853      base-fix .bool_set:N = \l__enumext_base_line_fix_bool,
854      base-fix .initial:n  = false,
855      base-fix .value_forbidden:n = true,
856    }
```

The function `\__enumext_nested_base_line_fix:` passed to the `\__enumext_parse_keys:n` function in the definition of the `enumext` environment (§13.38) will be responsible for applying the *baseline correction* and adjusting the ⟨*keys*⟩ for the `enumext` environment and the `\printkeyans` with *starred argument* '`*`' (§13.46).

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.

```
857 \cs_new_protected:Nn \__enumext_nested_base_line_fix:
858    {
859      \bool_lazy_all:nT
860        {
861          { \bool_if_p:N \l__enumext_starred_first_bool }
862          { \bool_if_p:N \l__enumext_base_line_fix_bool }
863          { \bool_not_p:n { \l__enumext_print_keyans_star_bool } }
864        }
865        {
866          \mode_leave_vertical:
```

```
867          \vspace { -\dim_eval:n { \baselineskip + \parsep } }
868        }
```

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`.

```
869      \bool_lazy_and:nnT
870        { \bool_if_p:N \l__enumext_starred_first_bool }
871        { \bool_if_p:N \l__enumext_print_keyans_star_bool }
872        {
873          \mode_leave_vertical:
874          \skip_vertical:n { -\baselineskip }
875          \skip_vertical:N \c_zero_skip
876        }
```

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.

```
877      \keys_set:nn { enumext / level-1 }
878        {
879          topsep = 0pt, above = 0pt, above* = 0pt,
880        }
881      \bool_set_false:N \l__enumext_base_line_fix_bool
882    }
```

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

13.17   Setting keys for horizontal spaces

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

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

```
906 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
907   {
908     \keys_define:nn { enumext / #1 }
909       {
910         itemindent    .dim_set:c = { l__enumext_fake_item_indent_#2_dim },
911         itemindent    .value_required:n = true,
912         rightmargin   .dim_set:c = { l__enumext_rightmargin_#2_dim },
913         rightmargin   .value_required:n = true,
914         listparindent .dim_set:c = { l__enumext_listparindent_#2_dim },
915         listparindent .value_required:n = true,
916         list-offset   .dim_set:c = { l__enumext_listoffset_#2_dim },
917         list-offset   .value_required:n = true,
918         list-indent   .meta:n    = { list-offset = ##1 },
919         list-indent   .value_required:n = true,
920       }
```

```
921        }
922    \clist_map_inline:nn
923      {
924        {enumext*}{vii}, {keyans*}{viii}
925      }
926      { \__enumext_tmp:nn #1 }
```

13.17.1   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.

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

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

13.18   Setting `show-length` key

show-length

Define and set `show-length` key for `enumext`, `enumext*`, `keyans` and `keyans*` environments. The function sets the boolean variable `\l__enumext_show_length_X_bool` used in the definition of all environments to

"true" and calls the function __enumext_show_length:nnn which prints all the values of the *"vertical"* and *"horizontal"* parameters calculated and used.

```
980 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
981   {
982     \keys_define:nn { enumext / #1 }
983       {
984         show-length .bool_set:c = { l__enumext_show_length_#2_bool },
985         show-length .initial:n  = false,
986       }
987   }
988 \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }
```

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

13.19   Setting before, after and first keys

<div style="float:left">before
before*
after
first</div>

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

```
989 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
990   {
991     \keys_define:nn { enumext / #1 }
992       {
993         before  .tl_set:c   = { l__enumext_before_no_starred_key_#2_tl },
994         before  .value_required:n = true,
995         before* .tl_set:c   = { l__enumext_before_starred_key_#2_tl },
996         before* .value_required:n = true,
997         after   .tl_set:c   = { l__enumext_after_stop_list_#2_tl },
998         after   .value_required:n = true,
999         first   .tl_set:c   = { l__enumext_after_list_args_#2_tl },
1000        first   .value_required:n = true,
1001      }
1002  }
1003 \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }
```

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

13.19.1   Functions for before, after and first keys in enumext

<div style="float:left">__enumext_before_args_exec:
__enumext_before_keys_exec:
__enumext_after_stop_list:
__enumext_after_args_exec:</div>

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*⟩}.

```
1004 \cs_new_protected:Nn \__enumext_before_args_exec:
1005   {
1006     \tl_use:c { l__enumext_before_starred_key_ \__enumext_level: _tl }
1007   }
```

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*⟩}}

```
1008 \cs_new_protected:Nn \__enumext_before_keys_exec:
1009   {
1010     \tl_use:c { l__enumext_before_no_starred_key_ \__enumext_level: _tl }
1011   }
```

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

```
1012 \cs_new_protected:Nn \__enumext_after_stop_list:
1013   {
1014     \tl_use:c { l__enumext_after_stop_list_ \__enumext_level: _tl }
1015   }
```

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.

```
1016 \cs_new_protected:Nn \__enumext_after_args_exec:
1017   {
1018     \tl_use:c { l__enumext_after_list_args_ \__enumext_level: _tl }
1019   }
```

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

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

__enumext_before_args_exec_v:
__enumext_before_keys_exec_v:
__enumext_after_stop_list_v:
__enumext_after_args_exec_v:

Same implementation as the one used in the enumext environment.

```
1020 \cs_new_protected:Nn \__enumext_before_args_exec_v:
1021   {
1022     \tl_use:N \l__enumext_before_starred_key_v_tl
1023   }
1024 \cs_new_protected:Nn \__enumext_before_keys_exec_v:
1025   {
1026     \tl_use:N \l__enumext_before_no_starred_key_v_tl
1027   }
1028 \cs_new_protected:Nn \__enumext_after_stop_list_v:
1029   {
1030     \tl_use:N \l__enumext_after_stop_list_v_tl
1031   }
1032 \cs_new_protected:Nn \__enumext_after_args_exec_v:
1033   {
1034     \tl_use:N \l__enumext_after_list_args_v_tl
1035   }
```

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

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

__enumext_before_args_exec_vii:
__enumext_before_keys_exec_vii
__enumext_after_stop_list_vii:
__enumext_after_args_exec_vii:

Same implementation as the one used in the enumext environment.

```
1036 \cs_new_protected:Nn \__enumext_before_args_exec_vii:
1037   {
1038     \tl_use:N \l__enumext_before_starred_key_vii_tl
1039   }
1040 \cs_new_protected:Nn \__enumext_before_args_exec_viii:
1041   {
1042     \tl_use:N \l__enumext_before_starred_key_viii_tl
1043   }
1044 \cs_new_protected:Nn \__enumext_before_keys_exec_vii:
1045   {
1046     \tl_use:N \l__enumext_before_no_starred_key_vii_tl
1047   }
1048 \cs_new_protected:Nn \__enumext_before_keys_exec_viii:
1049   {
1050     \tl_use:N \l__enumext_before_no_starred_key_viii_tl
1051   }
1052 \cs_new_protected:Nn \__enumext_after_stop_list_vii:
1053   {
1054     \tl_use:N \l__enumext_after_stop_list_vii_tl
1055   }
1056 \cs_new_protected:Nn \__enumext_after_stop_list_viii:
1057   {
1058     \tl_use:N \l__enumext_after_stop_list_viii_tl
1059   }
1060 \cs_new_protected:Nn \__enumext_after_args_exec_vii:
1061   {
1062     \tl_use:N \l__enumext_after_list_args_vii_tl
1063   }
1064 \cs_new_protected:Nn \__enumext_after_args_exec_viii:
1065   {
1066     \tl_use:N \l__enumext_after_list_args_viii_tl
1067   }
```

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

13.20 Setting keys for `multicols` and `minipage`

mini-env
mini-sep
columns-sep
columns

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

```
1068 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1069   {
1070     \keys_define:nn { enumext / #1 }
1071       {
1072         mini-env    .dim_set:c  = { l__enumext_minipage_right_#2_dim },
1073         mini-env    .value_required:n = true,
1074         mini-sep    .dim_set:c  = { l__enumext_minipage_hsep_#2_dim },
```

```
1075        mini-sep    .initial:n  = 0.3333em,
1076        mini-sep    .value_required:n = true,
1077        columns-sep .dim_set:c  = { l__enumext_columns_sep_#2_dim },
1078        columns-sep .value_required:n = true,
1079        columns     .int_set:c  = { l__enumext_columns_#2_int },
1080        columns     .initial:n  = 1,
1081        columns     .value_required:n = true,
1082      }
1083    }
1084 \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.

```
1085 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1086   {
1087     \keys_define:nn { enumext / #1 }
1088       {
1089         mini-right  .tl_gset:c = { g__enumext_miniright_code_#2_tl },
1090         mini-right  .value_required:n = true,
1091         mini-right* .code:n    = {
1092                                 \bool_gset_true:c { g__enumext_minipage_center_#2_bool }
1093                                 \keys_set:nn { enumext / #1 } { mini-right = {##1} }
1094                               },
1095         mini-right* .value_required:n = true,
1096       }
1097   }
1098 \clist_map_inline:nn { {enumext*}{vii}, {keyans*}{viii} } { \__enumext_tmp:nn #1 }
```

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

13.21 Adjustment of vertical spaces for `multicols`

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

Figure 7: Representation of the vertical space in `multicols` for a nested level.

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

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

13.21.1 Adjustment of vertical spaces for `multicols` in enumext

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

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

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

```
1099 \cs_new_protected:Nn \__enumext_multi_set_vskip:
1100   {
1101     \skip_set:cn { l__enumext_multicols_above_ \__enumext_level: _skip }
1102       {
1103         \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
1104       }
1105     \skip_set:cn { l__enumext_multicols_below_ \__enumext_level: _skip }
1106       {
1107         \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
1108       }
```

```
1109        \__enumext_add_pre_parsep:
1110      }
```

(*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*.

```
1111  \cs_new_protected:Nn \__enumext_add_pre_parsep:
1112    {
1113      \int_case:nn { \l__enumext_level_int }
1114        {
1115          { 2 }{
1116                \skip_if_eq:nnF { \l__enumext_parsep_i_skip } { \c_zero_skip }
1117                  {
1118                    \skip_add:Nn \l__enumext_multicols_above_ii_skip
1119                      {
1120                        \l__enumext_parsep_i_skip
1121                      }
1122                  }
1123            }
1124          { 3 }{
1125                \skip_if_eq:nnF { \l__enumext_parsep_ii_skip } { \c_zero_skip }
1126                  {
1127                    \skip_add:Nn \l__enumext_multicols_above_iii_skip
1128                      {
1129                        \l__enumext_parsep_ii_skip
1130                      }
1131                  }
1132            }
1133          { 4 }{
1134                \skip_if_eq:nnF { \l__enumext_parsep_iii_skip } { \c_zero_skip }
1135                  {
1136                    \skip_add:Nn \l__enumext_multicols_above_iv_skip
1137                      {
1138                        \l__enumext_parsep_iii_skip
1139                      }
1140                  }
1141            }
1142        }
1143    }
```

(*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*⟩.

```
1144  \cs_new_protected:Nn \__enumext_multi_addvspace:
1145    {
1146      \__enumext_multi_set_vskip:
1147      \mode_if_vertical:T
1148        {
1149          \skip_add:cn { l__enumext_multicols_above_ \__enumext_level: _skip }
1150            {
1151              \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
1152            }
1153          \skip_add:cn { l__enumext_multicols_below_ \__enumext_level: _skip }
1154            {
1155              \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
1156            }
1157        }
1158      \par\nopagebreak
1159      \addvspace{ \skip_use:c { l__enumext_multicols_above_ \__enumext_level: _skip } }
1160    }
```

(*End of definition for* __enumext_multi_addvspace:.)

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

`\__enumext_keyans_multi_set_vskip:`
`\__enumext_keyans_multi_addvspace:`

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

```
1161  \cs_new_protected:Nn \__enumext_keyans_multi_set_vskip:
1162    {
1163      \skip_set:Nn \l__enumext_multicols_above_v_skip
1164        {
1165          \l__enumext_topsep_v_skip
1166        }
1167      \skip_set:Nn \l__enumext_multicols_below_v_skip
1168        {
1169          \l__enumext_topsep_v_skip
1170        }
1171    }
1172  \cs_new_protected:Nn \__enumext_keyans_multi_addvspace:
1173    {
1174      \__enumext_keyans_multi_set_vskip:
1175      \mode_if_vertical:T
1176        {
1177          \skip_add:Nn \l__enumext_multicols_above_v_skip
1178            {
1179              \skip_use:N \l__enumext_partopsep_v_skip
1180            }
1181          \skip_add:Nn \l__enumext_multicols_below_v_skip
1182            {
1183              \skip_use:N \l__enumext_partopsep_v_skip
1184            }
1185        }
1186      \par\nopagebreak
1187      \addvspace{ \l__enumext_multicols_above_v_skip }
1188    }
```

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

13.22   Adjustment of vertical spaces for `minipage`

When nesting a *"list environment"* within the `minipage` environment, the values of the *"vertical spaces"* are lost. Graphically it can be seen like in the figure 8.

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

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

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

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

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

`\__enumext_minipage_set_skip:`
`\__enumext_minipage_add_space:`

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

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

```
1189  \cs_new_protected:Nn \__enumext_minipage_set_skip:
1190    {
1191      \skip_set:Nn \l__enumext_minipage_right_skip
```

```
1192        {
1193            \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
1194        }
1195      \mode_if_vertical:T
1196        {
1197            \skip_add:Nn \l__enumext_minipage_right_skip
1198              {
1199                \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
1200              }
1201        }
1202      \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:.

```
1203      \skip_set_eq:cN
1204        { l__enumext_multicols_above_ \__enumext_level: _skip } \l__enumext_minipage_right_skip
1205      \skip_set_eq:cN
1206        { l__enumext_multicols_below_ \__enumext_level: _skip } \l__enumext_minipage_right_skip
1207      \__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.

```
1208      \int_compare:nNnT
1209        { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } > { 1 }
1210        {
1211            \skip_zero:N \topskip
1212            \skip_set_eq:Nc \multicolsep { l__enumext_multicols_above_ \__enumext_level: _skip }
1213        }
1214    }
```

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*.

```
1215  \cs_new_protected:Nn \__enumext_minipage_add_space:
1216    {
1217      \__enumext_minipage_set_skip:
1218      \__enumext_unskip_unkern:
1219      \mode_if_vertical:TF
1220        {
1221            \nopagebreak\nointerlineskip
1222        }
1223        {
1224            \par\nopagebreak\nointerlineskip
1225            \skip_zero:c { l__enumext_partopsep_ \__enumext_level: _skip }
1226        }
1227      \int_compare:nNnTF
1228        { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } > { 1 }
1229        {
1230            \addvspace{ 0.445\box_ht:N \strutbox }
1231        }
1232        {
1233            \addvspace{ 0.250\box_ht:N \strutbox }
1234        }
1235    }
```

(*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.

```
1236  \cs_new_protected:Nn \__enumext_pre_itemsep_skip:
1237    {
1238      \int_case:nn { \l__enumext_level_int }
1239        {
1240            { 2 }{
1241                \skip_if_eq:nnTF
1242                  { \l__enumext_itemsep_i_skip } { \l__enumext_minipage_after_skip }
1243                  {
1244                    \skip_set:Nn \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1245                    \skip_set:Nn \l__enumext_multicols_below_ii_skip { 0.350\box_ht:N \strutbox }
```

```
1246                        }
1247                        {
1248                          \dim_compare:nNnT
1249                            { \l__enumext_itemsep_i_skip } < { \l__enumext_minipage_after_skip }
1250                            {
1251                              \skip_sub:Nn
1252                                \l__enumext_minipage_after_skip { \l__enumext_itemsep_i_skip }
1253                              \skip_sub:Nn
1254                                \l__enumext_multicols_below_ii_skip { \l__enumext_itemsep_i_skip }
1255                              \skip_add:Nn
1256                                \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1257                              \skip_add:Nn
1258                                \l__enumext_multicols_below_ii_skip { 0.350\box_ht:N \strutbox }
1259                            }
1260                          \dim_compare:nNnT
1261                            { \l__enumext_itemsep_i_skip } > { \l__enumext_minipage_after_skip }
1262                            {
1263                              \skip_set:Nn \l__enumext_minipage_temp_skip
1264                                {
1265                                  \l__enumext_itemsep_i_skip - \l__enumext_minipage_after_skip
1266                                }
1267                              \skip_sub:Nn
1268                                \l__enumext_minipage_after_skip { \l__enumext_itemsep_i_skip }
1269                              \skip_sub:Nn
1270                                \l__enumext_multicols_below_ii_skip { \l__enumext_itemsep_i_skip }
1271                              \skip_add:Nn
1272                                \l__enumext_minipage_after_skip
1273                                { 0.150\box_ht:N \strutbox + \l__enumext_minipage_temp_skip }
1274                              \skip_add:Nn
1275                                \l__enumext_multicols_below_ii_skip
1276                                { 0.350\box_ht:N \strutbox + \l__enumext_minipage_temp_skip }
1277                            }
1278                        }
1279                    }
1280          { 3 }{
1281                \skip_if_eq:nnTF
1282                    { \l__enumext_itemsep_ii_skip } { \c_zero_skip }
1283                    {
1284                      \skip_set:Nn \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1285                      \skip_set:Nn \l__enumext_multicols_below_iii_skip { 0.350\box_ht:N \strutbox }
1286                    }
1287                    {
1288                      \dim_compare:nNnT
1289                        { \l__enumext_itemsep_ii_skip } < { \l__enumext_minipage_after_skip }
1290                        {
1291                          \skip_sub:Nn
1292                            \l__enumext_minipage_after_skip { \l__enumext_itemsep_ii_skip }
1293                          \skip_sub:Nn
1294                            \l__enumext_multicols_below_iii_skip { \l__enumext_itemsep_ii_skip }
1295                          \skip_add:Nn
1296                            \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1297                          \skip_add:Nn
1298                            \l__enumext_multicols_below_iii_skip { 0.350\box_ht:N \strutbox }
1299                        }
1300                      \dim_compare:nNnT
1301                        { \l__enumext_itemsep_ii_skip } > { \l__enumext_minipage_after_skip }
1302                        {
1303                          \skip_set:Nn \l__enumext_minipage_temp_skip
1304                            {
1305                              \l__enumext_itemsep_ii_skip - \l__enumext_minipage_after_skip
1306                            }
1307                          \skip_sub:Nn
1308                            \l__enumext_minipage_after_skip { \l__enumext_itemsep_ii_skip }
1309                          \skip_sub:Nn
1310                            \l__enumext_multicols_below_iii_skip { \l__enumext_itemsep_ii_skip }
1311                          \skip_add:Nn
1312                            \l__enumext_minipage_after_skip
1313                            { 0.150\box_ht:N \strutbox + \l__enumext_minipage_temp_skip }
1314                          \skip_add:Nn
1315                            \l__enumext_multicols_below_iii_skip
1316                            { 0.350\box_ht:N \strutbox + \l__enumext_minipage_temp_skip }
```

```
1317                              }
1318                          }
1319                      }
1320              { 4 }{
1321                  \skip_if_eq:nnTF { \l__enumext_itemsep_iii_skip } { \c_zero_skip }
1322                      {
1323                          \skip_set:Nn \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1324                          \skip_set:Nn \l__enumext_multicols_below_iv_skip { 0.350\box_ht:N \strutbox }
1325                      }
1326                      {
1327                          \dim_compare:nNnT
1328                              { \l__enumext_itemsep_iii_skip } < { \l__enumext_minipage_after_skip }
1329                              {
1330                                  \skip_sub:Nn
1331                                      \l__enumext_minipage_after_skip { \l__enumext_itemsep_iii_skip }
1332                                  \skip_sub:Nn
1333                                      \l__enumext_multicols_below_iv_skip { \l__enumext_itemsep_iii_skip }
1334                                  \skip_add:Nn
1335                                      \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1336                                  \skip_add:Nn
1337                                      \l__enumext_multicols_below_iv_skip { 0.350\box_ht:N \strutbox }
1338                              }
1339                          \dim_compare:nNnT
1340                              { \l__enumext_itemsep_iii_skip } > { \l__enumext_minipage_after_skip }
1341                              {
1342                                  \skip_set:Nn \l__enumext_minipage_temp_skip
1343                                      {
1344                                          \l__enumext_itemsep_iii_skip - \l__enumext_minipage_after_skip
1345                                      }
1346                                  \skip_sub:Nn
1347                                      \l__enumext_minipage_after_skip { \l__enumext_itemsep_iii_skip }
1348                                  \skip_sub:Nn
1349                                      \l__enumext_multicols_below_iv_skip { \l__enumext_itemsep_iii_skip }
1350                                  \skip_add:Nn
1351                                      \l__enumext_minipage_after_skip
1352                                      { 0.150\box_ht:N \strutbox + \l__enumext_minipage_temp_skip }
1353                                  \skip_add:Nn
1354                                      \l__enumext_multicols_below_iv_skip
1355                                      { 0.350\box_ht:N \strutbox + \l__enumext_minipage_temp_skip }
1356                              }
1357                      }
1358                  }
1359              }
1360          }
```

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

13.22.2 Adjustment of vertical spaces for `minipage` in keyans

__enumext_keyans_minipage_set_skip:
__enumext_keyans_minipage_add_space:
__enumext_keyans_pre_itemsep_skip:

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

```
1361  \cs_new_protected:Nn \__enumext_keyans_minipage_set_skip:
1362    {
1363      \skip_zero:N \l__enumext_minipage_after_skip
1364      \skip_zero:N \l__enumext_minipage_left_skip
1365      \skip_zero:N \l__enumext_minipage_right_skip
1366      \skip_set:Nn \l__enumext_minipage_right_skip
1367        {
1368          \l__enumext_topsep_v_skip
1369        }
1370      \mode_if_vertical:T
1371        {
1372          \skip_add:Nn \l__enumext_minipage_right_skip
1373            {
1374              \l__enumext_partopsep_v_skip
1375            }
1376        }
1377      \skip_set_eq:NN \l__enumext_minipage_after_skip \l__enumext_minipage_right_skip
1378      \skip_set_eq:NN \l__enumext_multicols_above_v_skip \l__enumext_minipage_right_skip
1379      \skip_set_eq:NN \l__enumext_multicols_below_v_skip \l__enumext_minipage_right_skip
1380      \__enumext_keyans_pre_itemsep_skip:
```

```
1381        \int_compare:nNnT { \l__enumext_columns_v_int } > { 1 }
1382          {
1383            \skip_zero:N \topskip
1384            \skip_set_eq:NN \multicolsep \l__enumext_minipage_right_skip
1385          }
1386      }
1387  \cs_new_protected:Nn \__enumext_keyans_minipage_add_space:
1388      {
1389        \__enumext_keyans_minipage_set_skip:
1390        \__enumext_unskip_unkern:
1391        \mode_if_vertical:TF
1392          {
1393            \nopagebreak\nointerlineskip
1394          }
1395          {
1396            \par\nopagebreak\nointerlineskip
1397            \skip_zero:N \l__enumext_partopsep_v_skip
1398          }
1399        \int_compare:nNnTF { \l__enumext_columns_v_int } > { 1 }
1400          {
1401            \addvspace{ 0.445\box_ht:N \strutbox }
1402          }
1403          {
1404            \addvspace{ 0.250\box_ht:N \strutbox }
1405          }
1406      }
1407  \cs_new_protected:Nn \__enumext_keyans_pre_itemsep_skip:
1408      {
1409        \skip_if_eq:nnTF
1410          { \l__enumext_itemsep_i_skip } { \l__enumext_minipage_after_skip }
1411          {
1412            \skip_set:Nn \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1413            \skip_set:Nn \l__enumext_multicols_below_v_skip { 0.350\box_ht:N \strutbox }
1414          }
1415          {
1416            \dim_compare:nNnT
1417              { \l__enumext_itemsep_i_skip } < { \l__enumext_minipage_after_skip }
1418              {
1419                \skip_sub:Nn \l__enumext_minipage_after_skip { \l__enumext_itemsep_i_skip }
1420                \skip_sub:Nn \l__enumext_multicols_below_v_skip { \l__enumext_itemsep_i_skip }
1421                \skip_add:Nn \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1422                \skip_add:Nn \l__enumext_multicols_below_v_skip { 0.350\box_ht:N \strutbox }
1423              }
1424            \dim_compare:nNnT
1425              { \l__enumext_itemsep_i_skip } > { \l__enumext_minipage_after_skip }
1426              {
1427                \skip_set:Nn \l__enumext_minipage_temp_skip
1428                  {
1429                    \l__enumext_itemsep_i_skip - \l__enumext_minipage_after_skip
1430                  }
1431                \skip_sub:Nn \l__enumext_minipage_after_skip { \l__enumext_itemsep_i_skip }
1432                \skip_sub:Nn \l__enumext_multicols_below_v_skip { \l__enumext_itemsep_i_skip }
1433                \skip_add:Nn \l__enumext_minipage_after_skip
1434                  { 0.150\box_ht:N \strutbox + \l__enumext_minipage_temp_skip }
1435                \skip_add:Nn \l__enumext_multicols_below_v_skip
1436                  { 0.350\box_ht:N \strutbox + \l__enumext_minipage_temp_skip }
1437              }
1438          }
1439      }
```

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

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

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*`.

```
1440  \cs_new_protected:Nn \__enumext_mini_set_vskip_vii:
1441      {
1442        \skip_zero_new:N \l__enumext_minipage_left_skip
1443        \skip_gzero_new:N \g__enumext_minipage_right_skip
```

```
1444    \skip_gzero_new:N \g__enumext_minipage_after_skip
1445    \skip_if_eq:nnTF { \l__enumext_topsep_vii_skip } { \c_zero_skip }
1446      {
1447        \skip_set:Nn \l__enumext_minipage_left_skip { 0.5\box_dp:N \strutbox }
1448        \skip_gset:Nn \g__enumext_minipage_right_skip { 0.325\box_dp:N \strutbox }
1449      }
1450      {
1451        \skip_set:Nn \l__enumext_minipage_left_skip { 0.5875\box_dp:N \strutbox }
1452        \skip_gset:Nn \g__enumext_minipage_right_skip
1453          {
1454            \l__enumext_topsep_vii_skip
1455          }
1456        \skip_gset:Nn \g__enumext_minipage_after_skip
1457          {
1458            0.325\box_dp:N \strutbox + \l__enumext_topsep_vii_skip
1459          }
1460      }
1461    }
1462 \cs_new_protected:Nn \__enumext_mini_set_vskip_viii:
1463   {
1464     \skip_zero_new:N \l__enumext_minipage_after_skip
1465     \skip_zero_new:N \l__enumext_minipage_left_skip
1466     \skip_zero_new:N \l__enumext_minipage_right_skip
1467     \skip_if_eq:nnTF { \l__enumext_topsep_viii_skip } { \c_zero_skip }
1468       {
1469         \skip_set:Nn \l__enumext_minipage_left_skip
1470           {
1471             0.5\box_dp:N \strutbox
1472           }
1473         \skip_set:Nn \l__enumext_minipage_right_skip
1474           {
1475             \l__enumext_partopsep_viii_skip
1476           }
1477         \skip_set:Nn \l__enumext_minipage_after_skip
1478           {
1479             1.6\box_dp:N \strutbox
1480           }
1481       }
1482       {
1483         \skip_set:Nn \l__enumext_minipage_left_skip
1484           {
1485             0.5875\box_dp:N \strutbox
1486           }
1487         \skip_set:Nn \l__enumext_minipage_right_skip
1488           {
1489             \l__enumext_topsep_viii_skip
1490           }
1491         \skip_set:Nn \l__enumext_minipage_after_skip
1492           {
1493             0.325\box_dp:N \strutbox + \l__enumext_topsep_viii_skip
1494           }
1495       }
1496   }
```

(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.

```
1497 \cs_new_protected:Nn \__enumext_mini_addvspace_vii:
1498   {
1499     \__enumext_mini_set_vskip_vii:
1500     \par\nopagebreak
1501     \addvspace { \l__enumext_minipage_left_skip }
1502   }
1503 \cs_new_protected:Nn \__enumext_mini_addvspace_viii:
1504   {
1505     \__enumext_mini_set_vskip_viii:
1506     \par\nopagebreak
```

```
1507        \addvspace { \l__enumext_minipage_left_skip }
1508    }
```

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

13.22.4 The command \miniright

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

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

```
1509    \NewDocumentCommand \miniright { s }
1510    {
1511        \int_compare:nNnT { \l__enumext_keyans_pic_level_int } = { 1 }
1512        {
1513            \msg_error:nnn { enumext } { wrong-miniright-place }
1514        }
1515        % outside
1516        \bool_lazy_and:nnT
1517            { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
1518            { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
1519            {
1520                \msg_error:nnn { enumext } { wrong-miniright-place }
1521            }
1522        % starred env
1523        \bool_if:NT \l__enumext_starred_bool
1524            {
1525                \msg_error:nnn { enumext } { wrong-miniright-starred }
1526            }
1527        \int_compare:nNnTF { \l__enumext_keyans_level_int } = { 1 }
1528            {
1529                \__enumext_keyans_mini_right_cmd:n {#1}
1530            }
1531            { \__enumext_mini_right_cmd:n {#1} }
1532    }
```

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

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

```
1533    \cs_new_protected:Npn \__enumext_mini_right_cmd:n #1
1534    {
1535        \dim_compare:nNnTF
1536            { \dim_use:c { l__enumext_minipage_right_ \__enumext_level: _dim } } > { \c_zero_dim }
1537            {
1538                \__enumext_multicols_stop:
1539                \int_compare:nNnT
1540                    { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } = { 1 }
1541                    {
1542                        \par\addvspace{ \l__enumext_minipage_after_skip }
1543                    }
1544                \end__enumext_mini_page
1545                \hfill
1546                \__enumext_mini_page{ \dim_use:c { l__enumext_minipage_right_ \__enumext_level: _dim } }
1547                    \par\nointerlineskip
1548                    \addvspace { \l__enumext_minipage_right_skip }
1549                    \bool_if:nF {#1}
1550                        {
1551                            \centering
1552                        }
1553                    \int_gzero:N \g__enumext_minipage_stat_int
1554            }
```

```
1555        { \msg_error:nnn { enumext } { wrong-miniright-use } }
1556    % paranoia
1557    \RenewDocumentCommand \miniright { s }
1558      {
1559        \msg_error:nn { enumext } { many-miniright-used }
1560      }
1561  }
```

(*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.

```
1562 \cs_new_protected:Npn \__enumext_keyans_mini_right_cmd:n #1
1563  {
1564    \dim_compare:nNnTF { \l__enumext_minipage_right_v_dim } > { \c_zero_dim }
1565      {
1566        \__enumext_keyans_multicols_stop:
1567        \int_compare:nNnT { \l__enumext_columns_v_int } = { 1 }
1568          {
1569            \par\addvspace{ \l__enumext_minipage_after_skip }
1570          }
1571        \end__enumext_mini_page
1572        \hfill
1573        \__enumext_mini_page{ \l__enumext_minipage_right_v_dim }
1574          \par\nointerlineskip
1575          \addvspace { \l__enumext_minipage_right_skip }
1576          \bool_if:nF {#1}
1577            {
1578              \centering
1579            }
1580          \int_gzero:N \g__enumext_minipage_stat_int
1581      }
1582    { \msg_error:nnn { enumext } { wrong-miniright-use } }
1583    % paranoia
1584    \RenewDocumentCommand \miniright { s }
1585      {
1586        \msg_error:nn { enumext } { many-miniright-used }
1587      }
1588  }
```

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

13.23 Setting above and below keys

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

above
above*
below
below*

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

```
1589 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1590  {
1591    \keys_define:nn { enumext / #1 }
1592      {
1593        above  .skip_set:c = { l__enumext_vspace_above_#2_skip },
1594        above  .value_required:n = true,
1595        above* .code:n    = \bool_set_true:c { l__enumext_vspace_a_star_#2_bool }
1596                            \keys_set:nn { enumext / #1 } { above = {##1} },
1597        above* .value_required:n = true,
1598        below  .skip_set:c = { l__enumext_vspace_below_#2_skip },
1599        below  .value_required:n = true,
1600        below* .code:n    = \bool_set_true:c { l__enumext_vspace_b_star_#2_bool }
1601                            \keys_set:nn { enumext / #1 } { below = {##1} },
1602        below* .value_required:n = true,
1603      }
1604  }
1605 \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }
```

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

13.23.1   Functions for above and below keys in enumext

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

```
1606  \cs_new_protected:Nn \__enumext_vspace_above:
1607    {
1608      \skip_if_eq:nnF
1609        { \skip_use:c { l__enumext_vspace_above_ \__enumext_level: _skip } } { \c_zero_skip }
1610        {
1611          \bool_if:cTF { l__enumext_vspace_a_star_ \__enumext_level: _bool }
1612            {
1613              \vspace*{ \skip_use:c { l__enumext_vspace_above_ \__enumext_level: _skip } }
1614            }
1615            {
1616              \vspace { \skip_use:c { l__enumext_vspace_above_ \__enumext_level: _skip } }
1617            }
1618        }
1619    }
```

(*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.

```
1620  \cs_new_protected:Nn \__enumext_vspace_below:
1621    {
1622      \skip_if_eq:nnF
1623        { \skip_use:c { l__enumext_vspace_below_ \__enumext_level: _skip } } { \c_zero_skip }
1624        {
1625          \bool_if:cTF { l__enumext_vspace_b_star_ \__enumext_level: _bool }
1626            {
1627              \vspace*{ \skip_use:c { l__enumext_vspace_below_ \__enumext_level: _skip } }
1628            }
1629            {
1630              \vspace { \skip_use:c { l__enumext_vspace_below_ \__enumext_level: _skip } }
1631            }
1632        }
1633    }
```

(*End of definition for* __enumext_vspace_below:.)

13.23.2   Functions for above and below keys in keyans

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

```
1634  \cs_new_protected:Nn \__enumext_vspace_above_v:
1635    {
1636      \skip_if_eq:nnF { \l__enumext_vspace_above_v_skip } { \c_zero_skip }
1637        {
1638          \bool_if:NTF \l__enumext_vspace_a_star_v_bool
1639            {
1640              \vspace*{ \l__enumext_vspace_above_v_skip }
1641            }
1642            { \vspace { \l__enumext_vspace_above_v_skip } }
1643        }
1644    }
```

(*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.

```
1645  \cs_new_protected:Nn \__enumext_vspace_below_v:
1646    {
1647      \skip_if_eq:nnF { \l__enumext_vspace_below_v_skip } { \c_zero_skip }
1648        {
1649          \bool_if:NTF \l__enumext_vspace_b_star_v_bool
1650            {
1651              \vspace*{ \l__enumext_vspace_below_v_skip }
1652            }
1653            { \vspace { \l__enumext_vspace_below_v_skip } }
1654        }
1655    }
```

(*End of definition for* __enumext_vspace_below_v:.)

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

__enumext_vspace_above_vii:
__enumext_vspace_above_viii:

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

```
1656 \cs_new_protected:Nn \__enumext_vspace_above_vii:
1657   {
1658     \skip_if_eq:nnF { \l__enumext_vspace_above_vii_skip } { \c_zero_skip }
1659       {
1660         \bool_if:NTF \l__enumext_vspace_a_star_vii_bool
1661           {
1662             \vspace*{ \l__enumext_vspace_above_vii_skip }
1663           }
1664           { \vspace { \l__enumext_vspace_above_vii_skip } }
1665       }
1666   }
1667 \cs_new_protected:Nn \__enumext_vspace_above_viii:
1668   {
1669     \skip_if_eq:nnF { \l__enumext_vspace_above_viii_skip } { \c_zero_skip }
1670       {
1671         \bool_if:NTF \l__enumext_vspace_a_star_viii_bool
1672           {
1673             \vspace*{ \l__enumext_vspace_above_viii_skip }
1674           }
1675           { \vspace { \l__enumext_vspace_above_viii_skip } }
1676       }
1677   }
```

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

__enumext_vspace_below_vii:
__enumext_vspace_below_viii:

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

```
1678 \cs_new_protected:Nn \__enumext_vspace_below_vii:
1679   {
1680     \skip_if_eq:nnF { \l__enumext_vspace_below_vii_skip } { \c_zero_skip }
1681       {
1682         \bool_if:NTF \l__enumext_vspace_b_star_vii_bool
1683           {
1684             \vspace*{ \l__enumext_vspace_below_vii_skip }
1685           }
1686           { \vspace { \l__enumext_vspace_below_vii_skip } }
1687       }
1688   }
1689 \cs_new_protected:Nn \__enumext_vspace_below_viii:
1690   {
1691     \skip_if_eq:nnF { \l__enumext_vspace_below_viii_skip } { \c_zero_skip }
1692       {
1693         \bool_if:NTF \l__enumext_vspace_b_star_viii_bool
1694           {
1695             \vspace*{ \l__enumext_vspace_below_viii_skip }
1696           }
1697           { \vspace { \l__enumext_vspace_below_viii_skip } }
1698       }
1699   }
```

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

13.24 Setting series, resume and resume* keys

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

series
resume
resume*

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

```
1700 \cs_set_protected:Npn \__enumext_tmp:n #1
1701   {
1702     \keys_define:nn { enumext / #1 }
1703       {
1704         series  .str_set:N = \l__enumext_series_str,
1705         series  .value_required:n = true,
1706         resume  .code:n = \__enumext_resume_series:n {##1},
1707         resume* .code:n = \__enumext_resume_starred:,
```

```
1708           resume* .value_forbidden:n = true,
1709         }
1710     }
1711   \clist_map_inline:nn { level-1, enumext* } { \__enumext_tmp:n {#1} }
```

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

13.24.1   Internal functions for series key

__enumext_filter_series:n
__enumext_filter_series_key:n
__enumext_filter_series_pair:nn

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

```
1712 \cs_new:Npn \__enumext_filter_series:n #1
1713   {
1714     \use:e
1715       {
1716         \keyval_parse:NNn
1717           \__enumext_filter_series_key:n
1718           \__enumext_filter_series_pair:nn {#1}
1719       }
1720   }
```

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.

```
1721 \cs_new:Npn \__enumext_filter_series_key:n #1
1722   {
1723     \str_case:nnF {#1}
1724       {
1725         { resume } {} { resume* } {} { base-fix } {}
1726       }
1727       { , { \exp_not:n {#1} } }
1728   }
```

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.

```
1729 \cs_new:Npn \__enumext_filter_series_pair:nn #1#2
1730   {
1731     \str_case:nnF {#1}
1732       {
1733         { series } {} { resume } {} { start } {}
1734         { start* } {}  { save-ans } {} { save-key } {}
1735       }
1736       { , { \exp_not:n {#1} } = { \exp_not:n {#2} } }
1737   }
```

(*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 (§13.38) and to the function __enumext_parse_keys_vii:n in the enumext* environment definition (§13.43).

```
1738 \cs_new_protected:Npn \__enumext_parse_series:n #1
1739   {
1740     \str_if_empty:NTF \l__enumext_series_str
1741       {
1742         \bool_if:NF \l__enumext_resume_active_bool
1743           {
1744             \__enumext_resume_last:n {#1}
1745           }
1746       }
1747       {
1748         \tl_gclear_new:c { g__enumext_series_ \l__enumext_series_str _tl }
1749         \tl_gset:ce { g__enumext_series_ \l__enumext_series_str _tl }
1750           { \__enumext_filter_series:n {#1} }
1751         \int_if_exist:cF { g__enumext_series_ \l__enumext_series_str _int }
1752           {
```

```
1753              \int_new:c { g__enumext_series_ \l__enumext_series_str _int }
1754            }
1755        }
1756    }
```

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.

```
1757 \cs_new_protected:Npn \__enumext_resume_last:n #1
1758    {
1759      \bool_if:NT \l__enumext_standar_first_bool
1760        {
1761          \tl_gclear:N \g__enumext_standar_series_tl
1762          \tl_gset:Ne \g__enumext_standar_series_tl { \__enumext_filter_series:n {#1} }
1763        }
1764      \bool_if:NT \l__enumext_starred_first_bool
1765        {
1766          \tl_gclear:N \g__enumext_starred_series_tl
1767          \tl_gset:Ne \g__enumext_starred_series_tl { \__enumext_filter_series:n {#1} }
1768        }
1769    }
```

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

13.24.2   Internal function to save counter value

__enumext_resume_save_counter:

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

💣 The variables \l__enumext_series_str and \l__enumext__resume_name_tl contain the same {⟨series name⟩} but are executed at different moments, the integer variable with \l__enumext_series_str sets the value when execute series={⟨series name⟩} and the integer variable with \l__enumext__resume_name_tl sets the subsequent values when use resume={⟨series name⟩}. This function is passed to the enumext environment definition (§13.38) and the enumext* environment definition (§13.43).

```
1770 \cs_new_protected:Nn \__enumext_resume_save_counter:
1771    {
1772      \bool_if:NT \g__enumext_standar_bool
1773        {
1774          \tl_if_empty:NF \l__enumext_series_str
1775            {
1776              \int_gset_eq:cN
1777                { g__enumext_series_ \l__enumext_series_str _int } \value{enumXi}
1778            }
1779          \tl_if_empty:NTF \l__enumext_resume_name_tl
1780            {
1781              \str_if_empty:NT \l__enumext_series_str
1782                {
1783                  \int_gset_eq:NN \g__enumext_resume_int \value{enumXi}
1784                }
1785            }
1786            {
1787              \int_if_exist:cT { g__enumext_series_ \l__enumext_resume_name_tl _int }
1788                {
1789                  \int_gset_eq:cN
1790                    { g__enumext_series_ \l__enumext_resume_name_tl _int } \value{enumXi}
1791                }
1792            }
1793          \int_if_exist:cT { g__enumext_resume_ \l__enumext_store_name_tl _int }
1794            {
1795              \int_gset_eq:cN
1796                { g__enumext_resume_ \l__enumext_store_name_tl _int } \value{enumXi}
1797            }
1798        }
1799      \bool_if:NT \g__enumext_starred_bool
1800        {
1801          \tl_if_empty:NF \l__enumext_series_str
1802            {
1803              \int_gset_eq:cN
1804                { g__enumext_series_ \l__enumext_series_str _int } \value{enumXvii}
```

```
1805                    }
1806                \tl_if_empty:NTF \l__enumext_resume_name_tl
1807                  {
1808                    \str_if_empty:NT \l__enumext_series_str
1809                      {
1810                        \int_gset_eq:NN \g__enumext_resume_vii_int \value{enumXvii}
1811                      }
1812                  }
1813                  {
1814                    \int_if_exist:cT { g__enumext_series_ \l__enumext_resume_name_tl _int }
1815                      {
1816                        \int_gset_eq:cN
1817                          { g__enumext_series_ \l__enumext_resume_name_tl _int } \value{enumXvii}
1818                      }
1819                  }
1820                \int_if_exist:cT { g__enumext_resume_ \l__enumext_store_name_tl _int }
1821                  {
1822                    \int_gset_eq:cN
1823                      { g__enumext_resume_ \l__enumext_store_name_tl _int } \value{enumXvii}
1824                  }
1825            }
1826        }
```

(*End of definition for* __enumext_resume_save_counter:.)

13.24.3 Internal functions for resume key

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

```
1827  \cs_new_protected:Npn \__enumext_resume_series:n #1
1828    {
1829      \tl_if_empty:nTF {#1}
1830        {
1831          \__enumext_resume_counter:n { }
1832        }
1833        {
1834          \tl_if_exist:cTF { g__enumext_series_ \tl_to_str:n {#1} _tl }
1835            {
1836              \__enumext_resume_counter:n {#1}
1837              \bool_if:NT \g__enumext_standar_bool
1838                {
1839                  \keys_set:nv { enumext / level-1 }
1840                    { g__enumext_series_ \tl_to_str:n {#1} _tl }
1841                }
1842              \bool_if:NT \g__enumext_starred_bool
1843                {
1844                  \keys_set:nv { enumext / enumext* }
1845                    { g__enumext_series_ \tl_to_str:n {#1} _tl }
1846                }
1847            }
1848            {
1849              \bool_if:NT \g__enumext_standar_bool
1850                {
1851                  \msg_error:nnn { enumext } { unknown-series } {#1}
1852                }
1853              \bool_if:NT \g__enumext_starred_bool
1854                {
1855                  \msg_error:nnn { enumext } { unknown-series } {#1}
1856                }
1857            }
1858        }
1859    }
```

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

__enumext_resume_counter:n The function __enumext_resume_counter:n will set the variable \l__enumext_resume_active_bool
__enumext_resume_counter: to true and pass the value of the key resume to the variable \l__enumext_series_name_tl which will
  __enumext_resume_counter_series:

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.

```
1860  \cs_new_protected:Npn \__enumext_resume_counter:n #1
1861    {
1862      \bool_set_true:N \l__enumext_resume_active_bool
1863      \tl_set:Nn \l__enumext_resume_name_tl {#1}
1864      \tl_if_empty:NTF \l__enumext_resume_name_tl
1865        {
1866          \__enumext_resume_counter:
1867        }
1868        {
1869          \__enumext_resume_counter_series:
1870        }
1871      \__enumext_resume_counter_save_ans:
1872    }
```

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.

```
1873  \cs_new_protected:Nn \__enumext_resume_counter:
1874    {
1875      \bool_if:NT \g__enumext_standar_bool
1876        {
1877          \int_gincr:N \g__enumext_resume_int
1878          \int_set_eq:NN \l__enumext_start_i_int \g__enumext_resume_int
1879        }
1880      \bool_if:NT \g__enumext_starred_bool
1881        {
1882          \int_gincr:N \g__enumext_resume_vii_int
1883          \int_set_eq:NN \l__enumext_start_vii_int \g__enumext_resume_vii_int
1884        }
1885    }
```

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.

```
1886  \cs_new_protected:Nn \__enumext_resume_counter_series:
1887    {
1888      \bool_if:NT \g__enumext_standar_bool
1889        {
1890          \int_set:Nn \l__enumext_start_i_int
1891            {
1892              \int_use:c { g__enumext_series_ \l__enumext_resume_name_tl _int } + 1
1893            }
1894        }
1895      \bool_if:NT \g__enumext_starred_bool
1896        {
1897          \int_set:Nn \l__enumext_start_vii_int
1898            {
1899              \int_use:c { g__enumext_series_ \l__enumext_resume_name_tl _int } + 1
1900            }
1901        }
1902    }
```

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.

```
1903  \cs_new_protected:Nn \__enumext_resume_counter_save_ans:
1904    {
1905      \bool_lazy_and:nnT
1906        { \bool_if_p:N \l__enumext_standar_first_bool }
1907        { \bool_if_p:N \l__enumext_store_active_bool }
1908        {
1909          \int_set:Nn \l__enumext_start_i_int
1910            {
1911              \int_use:c { g__enumext_resume_ \l__enumext_store_name_tl _int } + 1
1912            }
1913        }
1914      \bool_lazy_and:nnT
```

```
1915        { \bool_if_p:N \l__enumext_starred_first_bool }
1916        { \bool_if_p:N \l__enumext_store_active_bool }
1917        {
1918          \int_set:Nn \l__enumext_start_vii_int
1919            {
1920              \int_use:c { g__enumext_resume_ \l__enumext_store_name_tl _int } + 1
1921            }
1922        }
1923    }
```

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

13.24.4 Internal function for resume* key

__enumext_resume_starred:

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

```
1924 \cs_new_protected:Nn \__enumext_resume_starred:
1925    {
1926      \bool_if:NT \g__enumext_standar_bool
1927        {
1928          \tl_if_empty:NF \g__enumext_standar_series_tl
1929            {
1930              \__enumext_resume_counter:n { }
1931              \keys_set:nV { enumext / level-1 } \g__enumext_standar_series_tl
1932            }
1933        }
1934      \bool_if:NT \g__enumext_starred_bool
1935        {
1936          \tl_if_empty:NF \g__enumext_starred_series_tl
1937            {
1938              \__enumext_resume_counter:n { }
1939              \keys_set:nV { enumext / enumext* } \g__enumext_starred_series_tl
1940            }
1941        }
1942    }
```

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

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

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

13.25.1 Setting save-ans key

save-ans

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

```
1943 \cs_set_protected:Npn \__enumext_tmp:n #1
1944    {
1945      \keys_define:nn { enumext / #1 }
1946        {
1947          save-ans .code:n = \__enumext_storing_set:n {##1},
1948          save-ans .value_required:n = true,
1949        }
1950    }
1951 \clist_map_inline:nn { level-1, enumext* } { \__enumext_tmp:n {#1} }
```

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

13.25.2 Internal functions for save-ans key

__enumext_start_save_ans_msg:
__enumext_stop_save_ans_msg:

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

```
1952 \cs_new_protected:Nn \__enumext_start_save_ans_msg:
1953    {
1954      \msg_term:nnVV { enumext } { save-ans-log }
1955        \g__enumext_envir_name_tl \l__enumext_store_name_tl
1956    }
1957 \cs_new_protected:Nn \__enumext_stop_save_ans_msg:
1958    {
```

```
1959        \msg_term:nnVV { enumext } { save-ans-log-hook }
1960           \g__enumext_envir_name_tl \g__enumext_store_name_tl
1961      }
```

(*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.

```
1962 \cs_new_protected:Npn \__enumext_storing_set:n #1
1963    {
1964      \tl_set:Ne \l__enumext_store_name_tl {#1}
1965      \tl_if_empty:NTF \l__enumext_store_name_tl
1966        {
1967          \bool_lazy_or:nnT
1968            { \l__enumext_standar_first_bool } { \l__enumext_starred_first_bool }
1969            {
1970              \msg_error:nnV { enumext } { save-ans-empty } \g__enumext_envir_name_tl
1971            }
1972        }
1973        {
1974          \bool_lazy_or:nnT
1975            { \l__enumext_standar_first_bool } { \l__enumext_starred_first_bool }
1976            {
1977              \__enumext_start_save_ans_msg:
1978              \__enumext_storing_exec:
1979            }
1980        }
1981    }
```

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 intenal 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* (§13.30).

```
1982 \cs_new_protected:Nn \__enumext_storing_exec:
1983    {
1984      \bool_set_true:N \l__enumext_store_active_bool
1985      \bool_set_true:N \l__enumext_check_answers_bool
1986      \tl_gset:NV \g__enumext_store_name_tl \l__enumext_store_name_tl
1987      \__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*.

```
1988        \prop_if_exist:cF { g__enumext_ \l__enumext_store_name_tl _prop }
1989          {
1990            \msg_log:nnV { enumext } { store-prop } \l__enumext_store_name_tl
1991            \prop_new:c { g__enumext_ \l__enumext_store_name_tl _prop }
1992          }
1993        \seq_if_exist:cF { g__enumext_ \l__enumext_store_name_tl _seq }
1994          {
1995            \msg_log:nnV { enumext } { store-seq } \l__enumext_store_name_tl
1996            \seq_new:c { g__enumext_ \l__enumext_store_name_tl _seq }
1997          }
1998        \int_if_exist:cF { g__enumext_resume_ \l__enumext_store_name_tl _int }
1999          {
2000            \msg_log:nnV { enumext } { store-int } \l__enumext_store_name_tl
2001            \int_new:c { g__enumext_resume_ \l__enumext_store_name_tl _int }
2002          }
2003    }
```

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

13.25.3 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.

13.25.4 Setting check-ans and no-store keys

check-ans
no-store

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

```
2004 \cs_set_protected:Npn \__enumext_tmp:n #1
2005   {
2006     \keys_define:nn { enumext / #1 }
2007       {
2008         check-ans .bool_set:N = \l__enumext_check_ans_key_bool,
2009         check-ans .initial:n  = false,
2010         check-ans .value_required:n = true,
2011         no-store  .code:n = {
2012                             \bool_set_false:N \l__enumext_check_answers_bool
2013                             \bool_set_false:N \l__enumext_check_ans_key_bool
2014                           },
2015         no-store  .value_forbidden:n = true,
2016       }
2017   }
2018 \clist_map_inline:nn
2019   {
2020     level-1, level-2, level-3, level-4, enumext*
2021   }
2022   { \__enumext_tmp:n {#1} }
```

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

13.25.5 Set-up check answer mechanism

__enumext_check_ans_active:
__enumext_check_ans_level:

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

```
2023 \cs_new_protected:Nn \__enumext_check_ans_active:
2024   {
2025     \tl_if_empty:NF \l__enumext_store_name_tl
2026       {
2027         \bool_if:NT \l__enumext_check_answers_bool
2028           {
2029             \__enumext_check_ans_level:
2030           }
2031       }
2032   }
```

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"*.

```
2033 \cs_new_protected:Nn \__enumext_check_ans_level:
2034   {
2035     \int_case:nn { \l__enumext_level_int }
2036       {
2037         { 1 }{
2038               \bool_lazy_all:nT
2039                 {
2040                   { \bool_if_p:N \g__enumext_starred_bool }
2041                   { \int_compare_p:nNn { \l__enumext_level_h_int } = { 1 } }
2042                 }
2043                 {
2044                   \int_gdecr:N \g__enumext_item_number_int
2045                   \bool_set_false:N \l__enumext_item_number_bool
2046                 }
2047             }
2048         { 2 }{
2049               \int_gdecr:N \g__enumext_item_number_int
2050               \bool_set_false:N \l__enumext_item_number_bool
2051             }
2052         { 3 }{
2053               \int_gdecr:N \g__enumext_item_number_int
2054               \bool_set_false:N \l__enumext_item_number_bool
2055             }
2056         { 4 }{
2057               \int_gdecr:N \g__enumext_item_number_int
2058               \bool_set_false:N \l__enumext_item_number_bool
2059             }
2060       }
```

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.

```
2061     \int_case:nn { \l__enumext_level_h_int }
2062       {
2063         { 1 }{
2064               \bool_lazy_all:nT
2065                 {
2066                   { \bool_if_p:N \g__enumext_standar_bool }
2067                   { \int_compare_p:nNn { \l__enumext_level_int } = { 1 } }
2068                 }
2069                 {
2070                   \int_gdecr:N \g__enumext_item_number_int
2071                   \bool_set_false:N \l__enumext_item_number_bool
2072                 }
2073             }
2074       }
2075   }
```

(*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.

```
2076 \cs_new_protected:Nn \__enumext_check_ans_key_hook:
2077   {
2078     \bool_lazy_and:nnT
2079       { \bool_if_p:N \l__enumext_check_ans_key_bool }
2080       { \bool_if_p:N \g__enumext_standar_bool }
2081       {
2082         \bool_gset_true:N \g__enumext_check_ans_key_bool
2083       }
2084     \bool_lazy_and:nnT
2085       { \bool_if_p:N \l__enumext_check_ans_key_bool }
2086       { \bool_if_p:N \g__enumext_starred_bool }
2087       {
2088         \bool_gset_true:N \g__enumext_check_ans_key_bool
2089       }
2090   }
```

(*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:.

```
2091 \cs_new_protected:Nn \__enumext_item_answer_diff:
2092   {
2093     \int_gset:Nn \g__enumext_item_answer_diff_int
2094       {
2095         \int_sign:n { \g__enumext_item_number_int - \g__enumext_item_anskey_int }
2096       }
2097   }
```

(*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:.

```
2098 \cs_new_protected:Nn \__enumext_check_ans_show:
2099   {
2100     \int_case:nn { \g__enumext_item_answer_diff_int }
2101       {
2102         { -1 }{ \__enumext_check_ans_msg_less:     }
2103         {  0 }{ \__enumext_check_ans_msg_same_ok: }
2104         {  1 }{ \__enumext_check_ans_msg_greater: }
2105       }
2106   }
2107 \cs_new_protected:Nn \__enumext_check_ans_msg_less:
2108   {
2109     \msg_warning:nneee { enumext } { item-less-answer } { \g__enumext_store_name_tl }
2110       { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2111   }
2112 \cs_new_protected:Nn \__enumext_check_ans_msg_same_ok:
2113   {
2114     \msg_term:nneee { enumext } { items-same-answer } { \g__enumext_store_name_tl }
2115       { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2116   }
2117 \cs_new_protected:Nn \__enumext_check_ans_msg_greater:
2118   {
2119     \msg_warning:nneee { enumext } { item-greater-answer } { \g__enumext_store_name_tl }
2120       { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2121   }
```

(*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:.

```
2122 \cs_new_protected:Nn \__enumext_check_ans_log:
2123   {
2124     \int_case:nn { \g__enumext_item_answer_diff_int }
2125       {
2126         { -1 }{ \__enumext_check_ans_log_msg_less:     }
2127         {  0 }{ \__enumext_check_ans_log_msg_same_ok: }
2128         {  1 }{ \__enumext_check_ans_log_msg_greater: }
2129       }
2130   }
2131 \cs_new_protected:Nn \__enumext_check_ans_log_msg_less:
2132   {
2133     \msg_log:nneee { enumext } { item-less-answer } { \g__enumext_store_name_tl }
2134       { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2135   }
2136 \cs_new_protected:Nn \__enumext_check_ans_log_msg_same_ok:
2137   {
2138     \msg_log:nneee { enumext } { items-same-answer }  { \g__enumext_store_name_tl }
2139       { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2140   }
```

```
2141 \cs_new_protected:Nn \__enumext_check_ans_log_msg_greater:
2142   {
2143     \msg_log:nneee { enumext } { item-greater-answer } { \g__enumext_store_name_tl }
2144       { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2145   }
```

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

13.25.6   Check for \item* and \anspic* commands

__enumext_check_starred_cmd:n

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

```
2146 \cs_new_protected:Npn \__enumext_check_starred_cmd:n #1
2147   {
2148     \int_compare:nNnT
2149       { \g__enumext_check_starred_cmd_int } = { 0 }
2150       {
2151         \msg_warning:nnnV
2152           { enumext } { missing-starred }{ #1 } \l__enumext_check_start_line_env_tl
2153       }
2154     \int_compare:nNnT
2155       { \g__enumext_check_starred_cmd_int } > { 1 }
2156       {
2157         \msg_warning:nnnV
2158           { enumext } { many-starred }{ #1 } \l__enumext_check_start_line_env_tl
2159       }
2160     \int_gzero:N \g__enumext_check_starred_cmd_int
2161     \tl_clear:N \l__enumext_check_start_line_env_tl
2162   }
```

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

13.26   Keys and functions associated with storage

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

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

```
2163 \cs_set_protected:Npn \__enumext_tmp:n #1
2164   {
2165     \keys_define:nn { enumext / #1 }
2166       {
2167         wrap-ans   .cs_set_protected:Np = \__enumext_anskey_wrapper:n ##1,
2168         wrap-ans   .initial:n =
2169                      {
2170                        \fbox{\parbox[t]{\dimeval{\itemwidth -2\fboxsep -2\fboxrule}}{##1}}
2171                      },
2172         wrap-ans   .value_required:n = true,
2173         wrap-opt   .cs_set_protected:Np = \__enumext_keyans_wrapper_opt:n ##1,
2174         wrap-opt   .initial:n = [{##1}],
2175         wrap-opt   .value_required:n = true,
2176         save-sep   .tl_set:N  = \l__enumext_store_keyans_item_opt_sep_tl,
2177         save-sep   .initial:n = {, ~ },
2178         save-sep   .value_required:n = true,
2179         mark-ans   .tl_set:N  = \l__enumext_mark_answer_sym_tl,
2180         mark-ans   .initial:n = \textasteriskcentered,
2181         mark-ans   .value_required:n = true,
2182         mark-pos   .choice:,
2183         mark-pos / left   .code:n = \str_set:Nn \l__enumext_mark_position_str { l },
2184         mark-pos / right  .code:n = \str_set:Nn \l__enumext_mark_position_str { r },
2185         mark-pos / unknown .code:n =
2186                          \msg_error:nneee { enumext } { unknown-choice }
2187                            { mark-pos } { left, ~ right } { \exp_not:n {##1} },
2188         mark-pos   .initial:n = right,
2189         mark-pos   .value_required:n = true,
2190         show-ans   .bool_set:N = \l__enumext_show_answer_bool,
2191         show-ans   .initial:n  = false,
2192         show-ans   .value_required:n = true,
2193         show-pos   .bool_set:N = \l__enumext_show_position_bool,
2194         show-pos   .initial:n  = false,
2195         show-pos   .value_required:n = true,
2196         mark-ref   .tl_set:N   = \l__enumext_mark_ref_sym_tl,
```

```
2197          mark-ref   .initial:n  = \textreferencemark,
2198          mark-ref   .value_required:n = true,
2199          save-ref   .bool_set:N = \l__enumext_store_ref_key_bool,
2200          save-ref   .initial:n  = false,
2201          save-ref   .value_required:n = true,
2202       }
2203    }
2204 \clist_map_inline:nn { level-1, enumext* } { \__enumext_tmp:n {#1} }
```

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

mark-pos
show-ans
show-pos

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

```
2205 \cs_set_protected:Npn \__enumext_tmp:n #1
2206    {
2207       \keys_define:nn { enumext / #1 }
2208          {
2209          mark-pos .choice:,
2210          mark-pos / left  .code:n = \str_set:Nn \l__enumext_mark_position_str { l },
2211          mark-pos / right .code:n = \str_set:Nn \l__enumext_mark_position_str { r },
2212          mark-pos .initial:n = right,
2213          mark-pos .value_required:n  = true,
2214          show-ans .bool_set:N = \l__enumext_show_answer_bool,
2215          show-ans .initial:n  = false,
2216          show-ans .value_required:n = true,
2217          show-pos .bool_set:N = \l__enumext_show_position_bool,
2218          show-pos .initial:n  = false,
2219          show-pos .value_required:n = true,
2220          }
2221    }
2222 \clist_map_inline:nn { keyans, keyans* } { \__enumext_tmp:n {#1} }
```

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

13.26.1 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*⟩.

```
2223 \cs_new_protected:Npn \__enumext_store_active_keys:n #1
2224    {
2225       \bool_if:cF { l__enumext_store_save_key_ \__enumext_level: _bool }
2226          {
2227          \tl_clear:c { l__enumext_save_key_ \__enumext_level: _tl }
2228          \tl_set:ce
2229             { l__enumext_store_save_key_ \__enumext_level: _tl }
2230             { \__enumext_filter_save_key:n {#1} }
2231          }
2232    }
2233 \cs_new_protected:Npn \__enumext_store_active_keys_vii:n #1
2234    {
2235       \bool_if:NF \l__enumext_store_save_key_vii_bool
2236          {
2237          \tl_clear:N \l__enumext_store_save_key_vii_tl
2238          \tl_set:Ne \l__enumext_store_save_key_vii_tl { \__enumext_filter_save_key:n {#1} }
2239          }
2240    }
```

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

13.26.2 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.

```
2241 \cs_set_protected:Npn \__enumext_tmp:n #1
2242   {
2243     \keys_define:nn { enumext / enumext* }
2244       {
2245         save-key .code:n = \__enumext_parse_save_key_vii:n {##1},
2246         save-key .value_required:n = true,
2247       }
2248     \keys_define:nn { enumext / #1 }
2249       {
2250         save-key .code:n = \__enumext_parse_save_key:n {##1},
2251         save-key .value_required:n = true,
2252       }
2253   }
2254 \clist_map_inline:nn { level-1, level-2, level-3, level-4 } { \__enumext_tmp:n {#1} }
```

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

__enumext_parse_save_key:n The functions __enumext_parse_save_key:n and __enumext_parse_save_key_vii:n will be respon-
__enumext_parse_save_key_vii:n sible for *"storing keys"* in the variable \l__enumext_store_save_key_X_tl for enumext and enumext*.

```
2255 \cs_new_protected:Npn \__enumext_parse_save_key:n #1
2256   {
2257     \bool_set_true:c { l__enumext_store_save_key_ \__enumext_level: _bool }
2258     \tl_clear:c { l__enumext_save_key_ \__enumext_level: _tl }
2259     \tl_set:ce
2260       { l__enumext_store_save_key_ \__enumext_level: _tl }
2261       { \__enumext_filter_save_key:n {#1} }
2262   }
2263 \cs_new_protected:Npn \__enumext_parse_save_key_vii:n #1
2264   {
2265     \bool_set_true:N \l__enumext_store_save_key_vii_bool
2266     \tl_clear:N \l__enumext_store_save_key_vii_tl
2267     \tl_set:Ne \l__enumext_store_save_key_vii_tl { \__enumext_filter_save_key:n {#1} }
2268   }
```

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

13.26.3 Internal functions to store optional arguments

__enumext_filter_save_key:n The function __enumext_filter_save_key:n will be in charge of *"filtering keys"* we want to *stored* in
__enumext_filter_save_key_key:n *sequence* where {#1} represents the *optional argument* passed to the environment.
__enumext_filter_save_key_pair:nn

```
2269 \cs_new:Npn \__enumext_filter_save_key:n #1
2270   {
2271     \use:e
2272       {
2273         \keyval_parse:NNn
2274           \__enumext_filter_save_key_key:n
2275           \__enumext_filter_save_key_pair:nn {#1}
2276       }
2277   }
```

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.

```
2278 \cs_new:Npn \__enumext_filter_save_key_key:n #1
2279   {
2280     \str_case:nnF {#1}
2281       {
2282         { resume } {} { resume* } {} { no-store } {} { base-fix } {}
2283       }
2284       { , { \exp_not:n {#1} } }
2285   }
```

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, check-ans, show-ans, save-pos, wrap-ans, mark-ans, wrap-opt, save-sep, mark-ref, mini-env, mini-sep, mini-right and mini-right* keys.

```
2286 \cs_new:Npn \__enumext_filter_save_key_pair:nn #1#2
2287   {
2288     \str_case:nnF {#1}
2289       {
2290         { series  } {} { resume   } {} { save-ans } {} { save-ref  } {}
2291         { save-key } {} { check-ans } {} { show-ans } {} { show-pos  } {}
2292         { wrap-ans } {} { mark-ans  } {} { wrap-opt } {} { save-sep  } {}
2293         { mark-ref } {} { mini-env  } {} { mini-sep } {} { mini-right } {}
2294         { mini-right* } {}
2295       }
2296     { , { \exp_not:n {#1} } = { \exp_not:n {#2} } }
2297   }
```

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

13.26.4 Function for storing content in prop list

__enumext_store_addto_prop:n
__enumext_store_addto_prop:V

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

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

```
2298 \cs_new_protected:Npn \__enumext_store_addto_prop:n #1
2299   {
2300     \prop_gput_if_not_in:cen { g__enumext_ \l__enumext_store_name_tl _prop }
2301       {
2302         \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } + 1 }
2303       }
2304     { #1 }
2305   }
2306 \cs_generate_variant:Nn \__enumext_store_addto_prop:n { V }
```

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

13.26.5 Function for storing content in sequence

__enumext_store_addto_seq:n
__enumext_store_addto_seq:v
__enumext_store_addto_seq:V

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

The form in which the {⟨*content*⟩} is stored in *sequence* is in a internal enumext or enumext* environments with the *"same structure"* in which the command was executed.
The *"stored content"* is retrieved by means of the \printkeyans command.

```
2307 \cs_new_protected:Npn \__enumext_store_addto_seq:n #1
2308   {
2309     \seq_gput_right:cn { g__enumext_ \l__enumext_store_name_tl _seq } { #1 }
2310   }
2311 \cs_generate_variant:Nn \__enumext_store_addto_seq:n { v, V }
```

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

13.26.6 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.

```
2312 \cs_new_protected:Nn \__enumext_store_level_open:
2313   {
2314     \bool_if:NT \l__enumext_check_answers_bool
2315       {
2316         \tl_if_empty:cTF { l__enumext_store_save_key_ \__enumext_level: _tl }
2317           {
2318             \__enumext_store_addto_seq:n
2319               {
2320                 \item \begin{enumext}
2321               }
2322           }
2323           {
2324             \tl_put_left:cn { l__enumext_store_save_key_ \__enumext_level: _tl }
2325               {
```

```
2326                    \item \begin{enumext} [
2327                 }
2328              \tl_put_right:cn { l__enumext_store_save_key_ \__enumext_level: _tl }
2329                 {
2330                    ]
2331                 }
2332              \__enumext_store_addto_seq:v { l__enumext_store_save_key_ \__enumext_level: _tl }
2333           }
2334        }
2335     }
2336  \cs_new_protected:Nn \__enumext_store_level_close:
2337     {
2338        \bool_if:NT \l__enumext_check_answers_bool
2339           {
2340              \__enumext_store_addto_seq:n { \end{enumext} }
2341           }
2342     }
```

(*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.

```
2343  \cs_new_protected:Nn \__enumext_store_level_open_vii:
2344     {
2345        \bool_if:NT \l__enumext_check_answers_bool
2346           {
2347              \tl_if_empty:NTF \l__enumext_store_save_key_vii_tl
2348                 {
2349                    \__enumext_store_addto_seq:n
2350                       {
2351                          \item \begin{enumext*}
2352                       }
2353                 }
2354                 {
2355                    \tl_put_left:Nn \l__enumext_store_save_key_vii_tl
2356                       {
2357                          \item \begin{enumext*}[
2358                       }
2359                    \tl_put_right:Nn \l__enumext_store_save_key_vii_tl
2360                       {
2361                          ]
2362                       }
2363                    \__enumext_store_addto_seq:V \l__enumext_store_save_key_vii_tl
2364                 }
2365           }
2366     }
2367  \cs_new_protected:Nn \__enumext_store_level_close_vii:
2368     {
2369        \bool_if:NT \l__enumext_check_answers_bool
2370           {
2371              \__enumext_store_addto_seq:n { \end{enumext*} }
2372           }
2373     }
```

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

13.26.7 Function for show marks and position

__enumext_print_keyans_box:NN
__enumext_print_keyans_box:cc

The function __enumext_print_keyans_box:NN print a box in the left margin with \l__enumext_mark_-answer_sym_tl used by the wrap-ans, show-ans and show-pos keys. The function takes two arguments:

#1: \l__enumext_labelwidth_X_dim
#2: \l__enumext_labelsep_X_dim

```
2374  \cs_new_protected:Nn \__enumext_print_keyans_box:NN
2375     {
2376        \mode_leave_vertical:
2377        \skip_horizontal:n { -\dim_use:N #2 }
2378        \makebox[0pt][ r ]
2379           {
2380              \makebox[ \dim_use:N #1 ][ \l__enumext_mark_position_str ]
2381                 {
2382                    \tl_use:N \l__enumext_mark_answer_sym_tl
```

```
2383                    }
2384              }
2385          \skip_horizontal:n { \dim_use:N #2 }
2386      }
2387  \cs_generate_variant:Nn \__enumext_print_keyans_box:NN { cc }
```

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

13.27 The internal label and ref

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

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

```
2388  \cs_new_protected:Nn \__enumext_store_internal_ref:
2389    {
2390      \cs_set_protected:Npn \__enumext_tmp:n ##1
2391        {
2392          \tl_set_eq:cc { l__enumext_label_copy_##1_tl } { l__enumext_label_##1_tl }
2393          \tl_reverse:c { l__enumext_label_copy_##1_tl }
2394          \tl_remove_once:cn { l__enumext_label_copy_##1_tl } { . }
2395          \tl_reverse:c { l__enumext_label_copy_##1_tl }
2396        }
2397      \clist_map_inline:nn { i, ii, iii, iv, vii } { \__enumext_tmp:n {##1} }
2398      \cs_set:Npn \__enumext_tmp:n ##1
2399        { . \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.

```
2400          \bool_lazy_all:nT
2401            {
2402              { \bool_if_p:N \g__enumext_starred_bool }
2403              { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
2404            }
2405            {
2406              \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2407                { \tl_use:N \l__enumext_label_copy_vii_tl }
2408            }
2409          \bool_lazy_all:nT
2410            {
2411              { \bool_not_p:n { \g__enumext_standar_bool } }
2412              { \bool_if_p:N \l__enumext_standar_bool }
2413              { \int_compare_p:nNn { \l__enumext_level_int } > { 0 } }
2414            }
2415            {
2416              \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2417                {
2418                  \tl_use:N \l__enumext_label_copy_vii_tl
2419                  \int_step_function:nnN { 1 } { \l__enumext_level_int } \__enumext_tmp:n
2420                }
2421            }
```

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.

```
2422          \bool_lazy_all:nT
2423            {
2424              { \bool_if_p:N \g__enumext_standar_bool }
2425              { \int_compare_p:nNn { \l__enumext_level_int } > { 0 } }
2426              { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
2427            }
2428            {
2429              \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2430                {
2431                  \tl_use:N \l__enumext_label_copy_i_tl
2432                  \int_step_function:nnN { 2 } { \l__enumext_level_int } \__enumext_tmp:n
2433                }
2434            }
2435          \cs_set:Npn \__enumext_tmp:n ##1
2436            { \tl_use:c { l__enumext_label_copy_ \int_to_roman:n {##1} _tl } . }
2437          \bool_lazy_all:nT
```

```
2438        {
2439          { \bool_if_p:N \g__enumext_standar_bool }
2440          { \bool_if_p:N \l__enumext_starred_bool }
2441          { \int_compare_p:nNn { \l__enumext_level_int } > { 0 } }
2442        }
2443        {
2444          \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2445            {
2446              \int_step_function:nnN { 1 } { \l__enumext_level_int } \__enumext_tmp:n
2447              \tl_use:N \l__enumext_label_copy_vii_tl
2448            }
2449        }
```

Now we set the variable \l__enumext_newlabel_arg_one_tl which will contain $\{\langle \text{store name} : \text{position} \rangle\}$.

```
2450        \tl_put_right:Ne \l__enumext_newlabel_arg_one_tl
2451          {
2452            \l__enumext_store_name_tl \c_colon_str
2453            \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } }
2454          }
```

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.

```
2455        \tl_put_right:Ne \l__enumext_write_aux_file_tl
2456          {
2457            \__enumext_newlabel:nn
2458              { \exp_not:V \l__enumext_newlabel_arg_one_tl }
2459              { \l__enumext_newlabel_arg_two_tl }
2460          }
2461      \l__enumext_write_aux_file_tl
2462    }
```

(*End of definition for* __enumext_store_internal_ref:.)

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

__enumext_store_anskey_code:n

The internal function __enumext_store_anskey_code:n first we pass the $\{\langle \text{argument} \rangle\}$ 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"* $\{\langle \text{argument} \rangle\}$.

```
2463 \cs_new_protected:Npn \__enumext_store_anskey_code:n #1
2464    {
2465      \int_gincr:N \g__enumext_item_anskey_int
2466      \__enumext_store_addto_prop:n {#1}
2467      \bool_if:NT \l__enumext_store_ref_key_bool
2468        {
2469          \__enumext_store_internal_ref:
2470        }
2471      \__enumext_anskey_show_wrap_left:n { #1 }
```

Now we start processing the $[\langle \text{key} = \text{val} \rangle]$ 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 $\langle \text{keys} \rangle$, if the break-col key is present and the command is running under enumext (not in enumext*) we will add \columnbreak and then \item.

```
2472      \tl_clear:N \l__enumext_store_anskey_arg_tl
2473      \bool_lazy_and:nnT
2474        { \bool_if_p:N \l__enumext_store_columns_break_bool }
2475        { \bool_not_p:n { \l__enumext_starred_bool } }
2476        {
2477          \tl_put_left:Nn \l__enumext_store_anskey_arg_tl { \columnbreak }
2478        }
2479      \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 ($\langle \text{number} \rangle$) to \l__enumext_store_anskey_arg_tl.

```
2480      \bool_lazy_and:nnT
2481        { \bool_not_p:n { \l__enumext_starred_bool } }
2482        { \int_compare_p:nNn { \l__enumext_store_item_join_int } > { 1 } }
2483        {
2484          \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2485            {
2486              ( \exp_not:V \l__enumext_store_item_join_int )
2487            }
2488        }
```

And now we will review the keys item-star, item-sym* and item-pos* and pass them to \l__enumext_-store_anskey_arg_tl along with the {⟨argument⟩} for \anskey or ⟨body⟩ for anskey*.

```
2489      \bool_if:NTF \l__enumext_store_item_star_bool
2490        {
2491          \tl_put_right:Nn \l__enumext_store_anskey_arg_tl { * }
2492          \tl_if_empty:NF \l__enumext_store_item_symbol_tl
2493            {
2494              \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2495                {
2496                  [ \exp_not:V \l__enumext_store_item_symbol_tl ]
2497                }
2498            }
2499          \dim_compare:nT
2500            {
2501              \l__enumext_store_item_symbol_sep_dim != \c_zero_dim
2502            }
2503            {
2504              \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2505                {
2506                  [ \exp_not:V \l__enumext_store_item_symbol_sep_dim ]
2507                }
2508            }
2509          \tl_put_right:Nn \l__enumext_store_anskey_arg_tl {#1}
2510        }
2511        {
2512          \tl_put_right:Nn \l__enumext_store_anskey_arg_tl {#1}
2513        }
```

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*.

```
2514      \bool_lazy_and:nnT
2515        { \bool_if_p:N \l__enumext_store_ref_key_bool }
2516        { \bool_if_p:N \l__enumext_hyperref_bool }
2517        {
2518          \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2519            {
2520              \hfill \exp_not:N \hyperlink { \exp_not:V \l__enumext_newlabel_arg_one_tl }
2521                { \exp_not:V \l__enumext_mark_ref_sym_tl }
2522            }
2523        }
2524      \__enumext_store_addto_seq:V \l__enumext_store_anskey_arg_tl
2525    }
```

(*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.

```
2526  \cs_new_protected:Npn \__enumext_anskey_show_wrap_arg:n #1
2527    {
2528      \par
2529      \bool_if:NTF \l__enumext_starred_bool
2530        {
2531          \__enumext_print_keyans_box:NN
2532            \l__enumext_labelwidth_vii_dim  \l__enumext_labelsep_vii_dim
2533        }
2534        {
2535          \__enumext_print_keyans_box:cc
2536            { l__enumext_labelwidth_ \__enumext_level: _dim }
2537            { l__enumext_labelsep_ \__enumext_level: _dim }
2538        }
2539      \__enumext_anskey_wrapper:n { #1 }
2540    }
```

(*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.

```
2541  \cs_new_protected:Npn \__enumext_anskey_show_wrap_left:n #1
2542    {
```

```
2543        \bool_if:NT \l__enumext_show_answer_bool
2544          {
2545            \__enumext_anskey_show_wrap_arg:n { #1 }
2546          }
2547        \bool_if:NT \l__enumext_show_position_bool
2548          {
2549            \tl_set:Ne \l__enumext_mark_answer_sym_tl
2550              {
2551                \group_begin:
2552                \exp_not:N \normalfont
2553                \exp_not:N \footnotesize [ \int_eval:n
2554                  {
2555                    \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
2556                  }
2557                  ]
2558                \group_end:
2559              }
2560            \__enumext_anskey_show_wrap_arg:n { #1 }
2561          }
2562      }
```

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

13.29   The command \anskey

Since we will be *"storing content"* in a `list` environment within *sequences* and can (more or less) manage the options passed to each level, it is necessary that we have a little more control over \item when storing.

The \anskey command will cover this point and give it similar behaviour to that of \item in the enumext and enumext* environments executed as follows \anskey[⟨*key* = *val*⟩]{⟨*content*⟩}.

__enumext_anskey_unknown:n
__enumext_anskey_unknown:nn

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

```
2563  \keys_define:nn { enumext / anskey }
2564    {
2565      break-col .bool_set:N = \l__enumext_store_columns_break_bool,
2566      break-col .default:n  = true,
2567      break-col .value_forbidden:n = true,
2568      item-join .int_set:N  = \l__enumext_store_item_join_int,
2569      item-join .value_required:n = true,
2570      item-star .bool_set:N = \l__enumext_store_item_star_bool,
2571      item-star .default:n  = true,
2572      item-star .value_forbidden:n = true,
2573      item-sym* .tl_set:N   = \l__enumext_store_item_symbol_tl,
2574      item-sym* .value_required:n = true,
2575      item-pos* .dim_set:N  = \l__enumext_store_item_symbol_sep_dim,
2576      item-pos* .value_required:n = true,
2577      unknown   .code:n     = { \__enumext_anskey_unknown:n {#1} },
2578    }
```

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.

```
2579  \cs_new_protected:Npn \__enumext_anskey_unknown:n #1
2580    {
2581      \exp_args:NV \__enumext_anskey_unknown:nn \l_keys_key_str {#1}
2582    }
2583  \cs_new_protected:Npn \__enumext_anskey_unknown:nn #1 #2
2584    {
2585      \tl_if_blank:nTF {#2}
2586        {
2587          \msg_error:nnn { enumext } { anskey-cmd-key-unknown } {#1}
2588        }
2589        {
2590          \msg_error:nnnn { enumext } { anskey-cmd-key-value-unknown } {#1} {#2}
2591        }
2592    }
```

(*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`.

```
2593  \NewDocumentCommand \anskey { o +m }
2594    {
2595      \__enumext_anskey_safe_outer:
2596      \group_begin:
2597        \bool_if:NT \l__enumext_check_answers_bool
2598          {
2599            \tl_if_novalue:nF {#1}
2600              {
2601                \keys_set:nn { enumext / anskey } {#1}
2602              }
2603            \tl_if_blank:nTF {#2}
2604              {
2605                \msg_error:nn { enumext } { anskey-empty-arg }
2606              }
2607              {
2608                \__enumext_anskey_safe_inner:
2609                \__enumext_store_anskey_code:n {#2}
2610              }
2611          }
2612      \group_end:
2613    }
```

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

13.29.1 Internal functions for the command

`\__enumext_anskey_safe_outer:`
`\__enumext_anskey_safe_inner:`

The `\__enumext_store_anskey_safe_outer:` function will return the appropriate messages when the command is executed outside the environment in which the save-ans key was activated.

```
2614  \cs_new_protected:Nn \__enumext_anskey_safe_outer:
2615    {
2616      \bool_if:NF \l__enumext_store_active_bool
2617        {
2618          \msg_error:nnnn { enumext } { anskey-wrong-place }{ anskey }{ enumext }
2619        }
2620      \int_compare:nNnT { \l__enumext_keyans_level_int } = { 1 }
2621        {
2622          \msg_error:nnnn { enumext } { command-wrong-place }{ anskey }{ keyans }
2623        }
2624      \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
2625        {
2626          \msg_error:nnnn { enumext } { command-wrong-place }{ anskey }{ keyans* }
2627        }
2628      \int_compare:nNnT { \l__enumext_keyans_pic_level_int } = { 1 }
2629        {
2630          \msg_error:nnnn { enumext } { command-wrong-place }{ anskey }{ keyanspic }
2631        }
2632    }
```

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.

```
2633  \cs_new_protected:Nn \__enumext_anskey_safe_inner:
2634    {
2635      \int_incr:N \l__enumext_anskey_level_int
2636      \int_compare:nNnT { \l__enumext_anskey_level_int } > { 1 }
2637        {
2638          \msg_error:nn { enumext } { anskey-nested }
2639        }
2640      \bool_if:NF \l__enumext_item_number_bool
2641        {
2642          \msg_error:nn { enumext } { anskey-unnumber-item }
2643        }
2644      \mode_if_math:T
2645        {
2646          \msg_error:nne { enumext } { anskey-math-mode } { \c_backslash_str anskey }
2647        }
2648    }
```

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

13.30 The environment anskey*

Managing *verbatim content* in an environment is quite complicated, I learned that when creating the scontents package, so to be able to have support at this point it is best to play a little with the internal code of scontents and *hooks*. Some considerations I should have here before implementing this:

- If some package, class or user has defined the environment with the same name somewhere in the document it would be a problem, you would not know what argument has been passed to store-env, if you are using the key print-env or the write-out key, sure, I can detect and modify it within the enumext and enumext* environments, but it would look strange not to have some keys available when running within these environments.
- A better (perhaps a bit paranoid) option is to define it within the environment in which the save-ans key is executed. and have it available only when that key is executed, here I would have absolute control of the ⟨*keys*⟩ and I make sure that write-out is not used, then using *hooks after* I undefine it and using *hook before* I check if it has been created by any package, class or user and I return a error, then the user will have to see how to solve the problem.

__enumext_undefine_anskey_env:

The function __enumext_undefine_anskey_env: will undefine the environment anskey* and will be passed to the function __enumext_execute_after_env: (§13.31) which is executed after the environment in which the key save-ans is active.

```
2649 \cs_new_protected:Nn \__enumext_undefine_anskey_env:
2650   {
2651     \cs_undefine:c { anskey* }
2652     \cs_undefine:c { endanskey* }
2653     \cs_undefine:c { __scontents_anskey*_env_begin: }
2654     \cs_undefine:c { __scontents_anskey*_env_end: }
2655   }
```

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

```
2656 \__enumext_before_env:nn { enumext }
2657   {
2658     \bool_lazy_and:nnT
2659       { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
2660       { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
2661       {
2662         \cs_if_free:cF { __scontents_anskey*_env_begin: }
2663           {
2664             \msg_error:nnn { enumext } { anskey-env-error } { anskey* }
2665           }
2666       }
2667   }
2668 \__enumext_before_env:nn { enumext* }
2669   {
2670     \bool_lazy_and:nnT
2671       { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
2672       { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
2673       {
2674         \cs_if_free:cF { __scontents_anskey*_env_begin: }
2675           {
2676             \msg_error:nnn { enumext } { anskey-env-error } { anskey* }
2677           }
2678       }
2679   }
```

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.

```
2680 \__enumext_before_env:nn { anskey* }
2681   {
2682     \int_compare:nNnT { \l__enumext_keyans_level_int } = { 1 }
2683       {
2684         \msg_error:nnn { enumext } { anskey-env-wrong }{ keyans }
2685       }
2686     \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
2687       {
2688         \msg_error:nnn { enumext } { anskey-env-wrong } { keyans* }
2689       }
2690     \int_compare:nNnT { \l__enumext_keyans_pic_level_int } = { 1 }
2691       {
2692         \msg_error:nnn { enumext } { anskey-env-wrong } { keyanspic }
2693       }
2694     \bool_if:NF \l__enumext_item_number_bool
2695       {
```

```
2696          \msg_error:nn { enumext } { anskey-unnumber-item }
2697        }
2698    \mode_if_math:T
2699      {
2700        \msg_error:nnn { enumext } { anskey-math-mode } { anskey* }
2701      }
2702  }
```

(*End of definition for* __enumext_undefine_anskey_env:.)

anskey*
__enumext_anskey_env_make:n
__enumext_anskey_env_make:V
__enumext_anskey_env_define_keys:
__enumext_rescan_anskey_env:n

The function __enumext_anskey_env_make:n creates the environment anskey* (*custom version* of scontents environment) by setting the initial keys store-env={⟨*store name*⟩} and print-env=false. To maintain the *scope* of the environment and that it is only active when the key save-ans is active we will pass this function to the function __enumext_storing_exec: (§13.25.1) and we will execute it only if the variable \l__enumext_anskey_env_bool is true, with this we prevent it from being executed again when the environment is nested and the key save-ans is active, which returns an error for part of the package scontents.

```
2703  \cs_new_protected:Npn \__enumext_anskey_env_make:n #1
2704    {
2705      \bool_if:NT \l__enumext_anskey_env_bool
2706        {
2707          \newenvsc{anskey*}[store-env=#1,print-env=false]
2708          \__enumext_anskey_env_exec:
2709        }
2710    }
2711  \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.

```
2712  \cs_new_protected:Nn \__enumext_anskey_env_define_keys:
2713    {
2714      \keys_define:nn { scontents / scontents }
2715        {
2716          break-col .bool_gset:N = \g__enumext_store_columns_break_bool,
2717          break-col .default:n   = true,
2718          break-col .value_forbidden:n = true,
2719          item-join .int_gset:N  = \g__enumext_store_item_join_int,
2720          item-join .value_required:n = true,
2721          item-star .bool_gset:N = \g__enumext_store_item_star_bool,
2722          item-star .default:n   = true,
2723          item-star .value_forbidden:n = true,
2724          item-sym* .tl_gset:N   = \g__enumext_store_item_symbol_tl,
2725          item-sym* .value_required:n = true,
2726          item-pos* .dim_gset:N  = \g__enumext_store_item_symbol_sep_dim,
2727          item-pos* .value_required:n = true,
2728          print-env .undefine:,
2729          store-env .undefine:,
2730          write-out .undefine:,
2731          unknown   .code:n      = { \__enumext_anskey_env_unknown:n {##1} },
2732        }
2733    }
```

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.

```
2734  \cs_new_protected:Npn \__enumext_anskey_env_unknown:n #1
2735    {
2736      \exp_args:NV \__enumext_anskey_env_unknown:nn \l_keys_key_str {#1}
2737    }
2738  \cs_new_protected:Npn \__enumext_anskey_env_unknown:nn #1#2
2739    {
2740      \tl_if_blank:nTF {#2}
2741        {
2742          \msg_error:nnn { enumext } { anskey-env-key-unknown } {#1}
2743        }
2744        {
2745          \msg_error:nnnn { enumext } { anskey-env-key-value-unknown } {#1} {#2}
2746        }
2747    }
```

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`.

```
2748 \cs_new_protected:Nn \__enumext_anskey_env_reset_keys:
2749   {
2750     \keys_define:nn { scontents / scontents }
2751       {
2752         break-col .undefine:,
2753         item-join .undefine:,
2754         item-star .undefine:,
2755         item-sym* .undefine:,
2756         item-pos* .undefine:,
2757         write-out .code:n    = {
2758                                 \bool_set_false:N \l__scontents_storing_bool
2759                                 \bool_set_true:N  \l__scontents_writing_bool
2760                                 \tl_set:Nn \l__scontents_fname_out_tl {##1}
2761                               },
2762         write-out .value_required:n = true,
2763         print-env .meta:nn   = { scontents } { print-env = ##1 },
2764         print-env .default:n = true,
2765         store-env .meta:nn   = { scontents } { store-env = ##1 },
2766         unknown   .code:n    = { \__scontents_parse_environment_keys:n {##1} },
2767       }
2768   }
```

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*.

```
2769 \cs_new_protected:Npn \__enumext_rescan_anskey_env:n #1
2770   {
2771     \group_begin:
2772       \int_set:Nn \tex_newlinechar:D { `\^^J }
2773       \__scontents_rescan_tokens:x
2774         {
2775           \endgroup % This assumes \catcode`\\=0... Things might go off otherwise.
2776           #1
2777         }
2778   }
```

(*End of definition for* anskey* *and others. This function is documented on page 13.*)

`\__enumext_anskey_env_exec:` The function `\__enumext_anskey_env_exec:` will be responsible for processing all the code necessary for the execution of the environment. The first thing will be to add our ⟨*keys*⟩.

```
2779 \cs_new_protected:Nn \__enumext_anskey_env_exec:
2780   {
2781     \__enumext_before_env:nn { anskey* }
2782       {
2783         \__enumext_anskey_env_define_keys:
2784       }
```

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.

```
2785     \hook_if_empty:nF {env/anskey*/after}
2786       {
2787         \hook_gremove_code:nn {env/anskey*/after} { * }
2788       }
2789     \__enumext_after_env:nn { anskey* }
2790       {
2791         \__enumext_anskey_env_save_keys:
2792         \tl_clear:N \l__enumext_store_anskey_env_tl
2793         \tl_clear:N \l__enumext_store_anskey_opt_tl
2794         \bool_if:NT \l__enumext_check_answers_bool
2795           {
2796             \tl_gset:Ne \l__enumext_store_anskey_env_tl
2797               {
2798                 \seq_item:ce { g__scontents_name_ \l__enumext_store_name_tl _seq } { -1 }
2799               }
2800             \regex_match:nVTF
2801               { ^\s* \z | ^\s* \u{c__scontents_hidden_space_str} \z }
2802               \l__enumext_store_anskey_env_tl
```

```
2803                        {
2804                            \msg_error:nn { enumext } { anskey-empty-arg }
2805                        }
2806                        {
2807                            \__enumext_anskey_env_store:
2808                        }
2809                    }
2810                \__enumext_anskey_env_clean_vars:
2811                \__enumext_anskey_env_reset_keys:
2812            }
2813        }
```

🖱 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.

```
2814  \cs_new_protected:Nn \__enumext_anskey_env_save_keys:
2815    {
2816      \bool_lazy_and:nnT
2817        { \bool_if_p:N \g__enumext_store_columns_break_bool }
2818        { \bool_not_p:n { \l__enumext_starred_bool } }
2819        {
2820          \tl_put_left:Ne \l__enumext_store_anskey_opt_tl { ,break-col, }
2821        }
```

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

```
2822      \bool_lazy_and:nnT
2823        { \bool_not_p:n { \l__enumext_starred_bool } }
2824        { \int_compare_p:nNn { \g__enumext_store_item_join_int } > { 1 } }
2825        {
2826          \tl_put_left::Ne \l__enumext_store_anskey_opt_tl
2827            {
2828              ,item-join = \exp_not:V \g__enumext_store_item_join_int,
2829            }
2830        }
```

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

```
2831      \bool_if:NT \g__enumext_store_item_star_bool
2832        {
2833          \tl_put_left:Ne \l__enumext_store_anskey_opt_tl
2834            {
2835              ,item-star,
2836            }
2837          \tl_if_empty:NF \g__enumext_store_item_symbol_tl
2838            {
2839              \tl_put_left:Ne \l__enumext_store_anskey_opt_tl
2840                {
2841                  ,item-sym* = \exp_not:V \g__enumext_store_item_symbol_tl,
2842                }
2843            }
2844          \dim_compare:nT
2845            {
2846              \g__enumext_store_item_symbol_sep_dim != \c_zero_dim
2847            }
2848            {
2849              \tl_put_left:Ne \l__enumext_store_anskey_opt_tl
2850                {
2851                  ,item-pos* = \exp_not:V \g__enumext_store_item_symbol_sep_dim,
2852                }
2853            }
2854        }
2855    }
```

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`.

```
2856  \cs_new_protected:Nn \__enumext_anskey_env_store:
```

```
2857    {
2858      \group_begin:
2859        \tl_if_empty:NTF \l__enumext_store_anskey_opt_tl
2860          {
2861            \exp_args:Ne
2862              \__enumext_store_anskey_code:n
2863                {
2864                  \__enumext_rescan_anskey_env:n { \l__enumext_store_anskey_env_tl }
2865                }
2866          }
2867          {
2868            \keys_set_known:nV { enumext / anskey } \l__enumext_store_anskey_opt_tl
2869            \exp_args:Ne
2870              \__enumext_store_anskey_code:n
2871                {
2872                  \__enumext_rescan_anskey_env:n { \l__enumext_store_anskey_env_tl }
2873                }
2874          }
2875      \group_end:
2876    }
```

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

```
2877  \cs_new_protected:Nn \__enumext_anskey_env_clean_vars:
2878    {
2879      \bool_gset_false:N \g__enumext_store_columns_break_bool
2880      \int_gzero:N        \g__enumext_store_item_join_int
2881      \bool_gset_false:N \g__enumext_store_item_star_bool
2882      \tl_gclear:N        \g__enumext_store_item_symbol_tl
2883      \dim_gzero:N        \g__enumext_store_item_symbol_sep_dim
2884    }
```

(*End of definition for* __enumext_anskey_env_save_keys: , __enumext_anskey_env_store: , *and* __enumext_anskey_env_-
clean_vars: .)

13.31   Executing anskey*, check-ans and write .log

__enumext_execute_after_env: The __enumext_execute_after_env: function will first return the appropriate message for the end of the environment in which the save-ans key is being executed, then call the __enumext_item_answer_diff: function and then will write the values of the global variables used to the .log file. If the key check-ans is active it will execute the function __enumext_check_ans_show: and show the result in the terminal, otherwise it will execute the function __enumext_check_ans_log: and write the results in the .log file, undefine the environment anskey* (§13.30) through the function __enumext_undefine_anskey_env: and finally we execute the function __enumext_reset_global_vars: returning the used variables to their original state.

```
2885  \cs_new_protected:Nn \__enumext_execute_after_env:
2886    {
2887      \int_compare:nNnT { \l__enumext_level_int } = { 0 }
2888        {
2889          \tl_if_empty:NF \g__enumext_store_name_tl
2890            {
2891              \__enumext_stop_save_ans_msg:
2892              \__enumext_item_answer_diff:
2893              \__enumext_log_global_vars:
2894              \__enumext_log_answer_vars:
2895              \bool_if:NTF \g__enumext_check_ans_key_bool
2896                {
2897                  \__enumext_check_ans_show:
2898                }
2899                { \__enumext_check_ans_log: }
2900              \__enumext_undefine_anskey_env:
2901            }
2902          \__enumext_reset_global_vars:
2903        }
2904    }
```

(*End of definition for* __enumext_execute_after_env: .)

🍏 This function is passed to the function __enumext_after_env:nn for the environments enumext (§13.38) and enumext* (§13.43) and it is executed only when the environments are not nested or at some level of these..

13.32 Common functions for `keyans`, `keyans*` and `keyanspic`

13.32.1 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`.

```
2905  \cs_new_protected:Npn \__enumext_keyans_addto_prop:n #1
2906    {
2907      \tl_clear:N \l__enumext_store_current_label_tl
2908      \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
2909        {
2910          \tl_put_right:Ne \l__enumext_store_current_label_tl { \l__enumext_label_vi_tl }
2911        }
2912        {
2913          \tl_put_right:Ne \l__enumext_store_current_label_tl { \l__enumext_label_v_tl }
2914        }
```

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

```
2915      \tl_if_novalue:nF { #1 }
2916        {
2917          \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_tl
2918            {
2919              \tl_put_right:Ne \l__enumext_store_current_label_tl
2920                {
2921                  \l__enumext_store_keyans_item_opt_sep_tl
2922                }
2923            }
2924          \tl_put_right:Ne \l__enumext_store_current_label_tl { #1 }
2925        }
2926      \__enumext_store_addto_prop:V \l__enumext_store_current_label_tl
2927    }
```

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

13.32.2 The save-ref key for `keyans`, `keyans*` and `keyanspic`

The *"internal label and ref"* system for the `keyans`, `keyans*` and `keyanspic` environments has *slight differences* with the one implemented for `\anskey` basically because in this environments the interest is in the current ⟨label⟩ for `\item*` and `\anspic*` with the ⟨contents⟩ of the *optional argument*. The mechanism defined here will allow to execute `\ref{`⟨*store name : position*⟩`}` and will return 1. (A).

`\__enumext_keyans_store_ref:`
`\__enumext_keyans_store_ref_aux_i:`
`\__enumext_keyans_store_ref_aux_ii:`

The function `\__enumext_keyans_store_ref:` handles the *"internal label and ref"* system used by the `save-ref` key for `\item*` and `\anspic*` commands. First we will create copies of the current ⟨labels⟩ and remove the dots "`.`" from them, we do not want to get double dots in references.

```
2928  \cs_new_protected:Nn \__enumext_keyans_store_ref:
2929    {
2930      \bool_if:NT \l__enumext_store_ref_key_bool
2931        {
2932          \cs_set_protected:Npn \__enumext_tmp:n ##1
2933            {
2934              \tl_set_eq:cc { l__enumext_label_copy_##1_tl } { l__enumext_label_##1_tl }
2935              \tl_reverse:c { l__enumext_label_copy_##1_tl }
2936              \tl_remove_once:cn { l__enumext_label_copy_##1_tl } { . }
2937              \tl_reverse:c { l__enumext_label_copy_##1_tl }
2938            }
2939          \clist_map_inline:nn { i, v, vi, vii, viii } { \__enumext_tmp:n {##1} }
2940          \__enumext_keyans_store_ref_aux_i:
2941        }
2942    }
```

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`.

```
2943  \cs_new_protected:Nn \__enumext_keyans_store_ref_aux_i:
2944    {
2945      \bool_if:NT \g__enumext_starred_bool
2946        {
2947          \tl_set_eq:NN \l__enumext_label_copy_i_tl \l__enumext_label_copy_vii_tl
2948        }
2949      \int_compare:nNnT { \l__enumext_keyans_pic_level_int } = { 1 }
```

```
2950        {
2951            \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2952                { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_vi_tl }
2953        }
2954    \int_compare:nNnT { \l__enumext_keyans_level_int } = { 1 }
2955        {
2956            \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2957                { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_v_tl }
2958        }
2959    \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
2960        {
2961            \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2962                { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_viii_tl }
2963        }
2964    \tl_put_right:Ne \l__enumext_newlabel_arg_one_tl
2965        {
2966            \l__enumext_store_name_tl \c_colon_str
2967            \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } }
2968        }
2969    \__enumext_keyans_store_ref_aux_ii:
2970    }
```

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.

```
2971 \cs_new_protected:Nn \__enumext_keyans_store_ref_aux_ii:
2972    {
2973        \tl_put_right:Ne \l__enumext_write_aux_file_tl
2974            {
2975                \__enumext_newlabel:nn
2976                    { \exp_not:V \l__enumext_newlabel_arg_one_tl }
2977                    { \l__enumext_newlabel_arg_two_tl }
2978            }
2979        \l__enumext_write_aux_file_tl
2980    }
```

(*End of definition for* `\__enumext_keyans_store_ref:`, `\__enumext_keyans_store_ref_aux_i:`, *and* `\__enumext_keyans_-store_ref_aux_ii:`.)

13.32.3 Storing content in sequence

`\__enumext_keyans_addto_seq:n`
`\__enumext_keyans_addto_seq_link:`

The function `\__enumext_keyans_addto_seq:n` will pass the contents of the current ⟨*label*⟩ `\l__-enumext_label_v_tl` for the keyans environment and the `\l__enumext_label_vi_tl` for the keyanspic environment when using `\item*` and `\anspic*`, followed by the ⟨*contents*⟩ of the *optional argument* of both commands to the `\l__enumext_store_current_label_tl` variable to the sequence defined by the save-ans key.

```
2981 \cs_new_protected:Npn \__enumext_keyans_addto_seq:n #1
2982    {
2983        \tl_clear:N \l__enumext_store_current_label_tl
2984        \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
2985            {
2986                \tl_put_right:Ne \l__enumext_store_current_label_tl { \item \l__enumext_label_vi_tl }
2987            }
2988            {
2989                \tl_put_right:Ne \l__enumext_store_current_label_tl { \item \l__enumext_label_v_tl }
2990            }
2991        \tl_if_novalue:nF { #1 }
2992            {
2993                \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_tl
2994                    {
2995                        \tl_put_right:Ne \l__enumext_store_current_label_tl
2996                            {
2997                                \l__enumext_store_keyans_item_opt_sep_tl
2998                            }
2999                    }
3000                \tl_put_right:Ne \l__enumext_store_current_label_tl { #1 }
3001            }
3002        \__enumext_keyans_addto_seq_link:
3003    }
```

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.

```
3004  \cs_new_protected:Nn \__enumext_keyans_addto_seq_link:
3005    {
3006      \bool_lazy_and:nnT
3007        { \bool_if_p:N \l__enumext_store_ref_key_bool }
3008        { \bool_if_p:N \l__enumext_hyperref_bool }
3009        {
3010          \tl_put_right:Ne \l__enumext_store_current_label_tl
3011            {
3012              \hfill \exp_not:N \hyperlink
3013                {
3014                  \exp_not:V \l__enumext_newlabel_arg_one_tl
3015                }
3016                { \exp_not:V \l__enumext_mark_ref_sym_tl }
3017            }
3018        }
3019      \__enumext_store_addto_seq:V \l__enumext_store_current_label_tl
3020      \bool_if:NT \l__enumext_check_answers_bool
3021        {
3022          \int_gincr:N \g__enumext_item_anskey_int
3023        }
3024    }
```

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

13.32.4 The show-ans and show-pos keys for keyans and keyanspic

The code is very similar to the \anskey code, but, if I change the order of the operations the counter off ⟨label⟩ are incorrect.

__enumext_keyans_show_left:n
__enumext_keyans_show_ans:
__enumext_keyans_show_pos:
__enumext_keyans_show_item_opt:

Common function to show *starred commands* \item* and ⟨position⟩ of stored content in *prop list* for keyans and keyanspic. Need add 1 to \g__enumext_⟨store name⟩_prop for show-pos key.

```
3025  \cs_new_protected:Npn \__enumext_keyans_show_left:n #1
3026    {
3027      \tl_if_novalue:nF { #1 }
3028        {
3029          \tl_set:Ne \l__enumext_store_current_opt_arg_tl { #1 }
3030        }
3031      \bool_if:NT \l__enumext_show_answer_bool
3032        {
3033          \__enumext_keyans_show_ans:
3034        }
3035      \bool_if:NT \l__enumext_show_position_bool
3036        {
3037          \__enumext_keyans_show_pos:
3038        }
3039    }
3040  \cs_new_protected:Nn \__enumext_keyans_show_item_opt:
3041    {
3042      \tl_if_empty:NF \l__enumext_store_current_opt_arg_tl
3043        {
3044          \bool_lazy_or:nnT
3045            { \bool_if_p:N \l__enumext_show_answer_bool }
3046            { \bool_if_p:N \l__enumext_show_position_bool }
3047            {
3048              \__enumext_keyans_wrapper_opt:n { \l__enumext_store_current_opt_arg_tl } \c_space_tl
3049            }
3050        }
3051    }
3052  \cs_new_protected:Nn \__enumext_keyans_show_ans:
3053    {
3054      \bool_if:NT \l__enumext_starred_bool
3055        {
3056          \dim_set_eq:NN \l__enumext_labelwidth_i_dim \l__enumext_labelwidth_vii_dim
3057          \dim_set_eq:NN \l__enumext_labelsep_i_dim \l__enumext_labelsep_vii_dim
3058        }
3059      \tl_put_left:Nn \l__enumext_label_v_tl
3060        {
3061          \__enumext_print_keyans_box:NN
3062            \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
3063        }
```

```
3064      }
3065  \cs_new_protected:Nn \__enumext_keyans_show_pos:
3066    {
3067      \bool_if:NT \l__enumext_starred_bool
3068        {
3069          \dim_set_eq:NN \l__enumext_labelwidth_i_dim \l__enumext_labelwidth_vii_dim
3070          \dim_set_eq:NN \l__enumext_labelsep_i_dim \l__enumext_labelsep_vii_dim
3071        }
3072      \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
3073        {
3074          \tl_set:Ne \l__enumext_mark_answer_sym_tl
3075            {
3076              \group_begin:
3077              \exp_not:N \normalfont
3078              \exp_not:N \footnotesize [ \int_eval:n
3079                {
3080                  \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
3081                }
3082                ]
3083              \group_end:
3084            }
3085        }
3086        {
3087          \tl_set:Ne \l__enumext_mark_answer_sym_tl
3088            {
3089              \group_begin:
3090              \exp_not:N \normalfont
3091              \exp_not:N \footnotesize [ \int_eval:n
3092                {
3093                  \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } + 1
3094                }
3095                ]
3096              \group_end:
3097            }
3098        }
3099      \tl_put_left:Nn \l__enumext_label_v_tl
3100        {
3101          \__enumext_print_keyans_box:NN
3102            \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
3103        }
3104    }
```

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

13.33 Redefining `\item` **and** `\makelabel` **in enumext**

Redefining the `\item` command is not as simple as I thought. This command works in conjunction with the `\makelabel` command so I have to redefine both of them, in addition to this, we will have to use a couple of *global* variables to pass the values from one command to the other.

The `\item` and `\item[`⟨*custom*⟩`]` commands work in the usual way on enumext and we will add `\item*`, `\item*[`⟨*symbol*⟩`]` and `\item*[`⟨*symbol*⟩`][`⟨*offset*⟩`]`.

`\__enumext_default_item:n` First we will see if the *optional argument* is present, if it is NOT present we will check the state of the variable `\l__enumext_check_answers_bool` set by the key `no-store`, set the boolean variable `\l__enumext_-wrap_label_X_bool` to "true" for the key `wrap-label` and execute `\__enumext_item_std:w` and the key `itemindent`, otherwise we will check the state of the boolean variable `\l__enumext_wrap_label_opt_-X_bool` set by the key `wrap-label*` and execute `\__enumext_item_std:w` with the *optional argument* and the key `itemindent`.

```
3105  \cs_new_protected:Npn \__enumext_default_item:n #1
3106    {
3107      \tl_if_novalue:nTF {#1}
3108        {
3109          \bool_if:NT \l__enumext_check_answers_bool
3110            {
3111              \int_gincr:N \g__enumext_item_number_int
3112              \bool_set_true:N \l__enumext_item_number_bool
3113            }
3114          \bool_set_true:c { l__enumext_wrap_label_ \__enumext_level: _bool }
3115          \__enumext_item_std:w \tl_use:c { l__enumext_fake_item_indent_ \__enumext_level: _tl }
3116        }
3117        {
```

```
3118        \bool_set_eq:cc
3119          { l__enumext_wrap_label_ \__enumext_level: _bool }
3120          { l__enumext_wrap_label_opt_ \__enumext_level: _bool }
3121        \__enumext_item_std:w [#1] \tl_use:c { l__enumext_fake_item_indent_ \__enumext_level: _tl
3122      }
3123    }
```

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

__enumext_starred_item:nn
__enumext_item_star_exec:

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

#1: \l__enumext_item_symbol_X_tl
#2: \l__enumext_item_symbol_sep_X_dim

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

```
3124  \cs_new_protected:Npn \__enumext_starred_item:nn #1 #2
3125    {
3126      \tl_if_novalue:nTF {#1}
3127        {
3128          \tl_gset_eq:Nc
3129            \g__enumext_item_symbol_aux_tl { l__enumext_item_symbol_ \__enumext_level: _tl }
3130        }
3131        {
3132          \tl_gset:Nn \g__enumext_item_symbol_aux_tl {#1}
3133        }
3134      \tl_if_novalue:nTF {#2}
3135        {
3136          \dim_set_eq:cc
3137            { l__enumext_item_symbol_sep_ \__enumext_level: _dim }
3138            { l__enumext_labelsep_ \__enumext_level: _dim }
3139        }
3140        {
3141          \dim_set:cn { l__enumext_item_symbol_sep_ \__enumext_level: _dim } {#2}
3142        }
3143      \bool_if:NT \l__enumext_check_answers_bool
3144        {
3145          \int_gincr:N \g__enumext_item_number_int
3146          \bool_set_true:N \l__enumext_item_number_bool
3147        }
3148      \bool_set_true:c { l__enumext_wrap_label_ \__enumext_level: _bool }
3149      \__enumext_item_std:w \tl_use:c { l__enumext_fake_item_indent_ \__enumext_level: _tl }
3150    }
```

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

```
3151  \cs_new_protected:Nn \__enumext_item_star_exec:
3152    {
3153      \tl_if_empty:cF { l__enumext_item_symbol_ \__enumext_level: _tl }
3154        {
3155          \mode_leave_vertical:
3156          \skip_horizontal:n { -\dim_use:c { l__enumext_item_symbol_sep_ \__enumext_level: _dim } }
3157          \hbox_overlap_left:n { \g__enumext_item_symbol_aux_tl }
3158          \skip_horizontal:n { \dim_use:c { l__enumext_item_symbol_sep_ \__enumext_level: _dim } }
3159        }
3160    }
```

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

__enumext_redefine_item:

The function __enumext_redefine_item: will redefine the \item command in the enumext environment adding \item*. This function are passed to __enumext_list_arg_two_X: used in the definition of the enumext environment (§13.38).

```
3161  \cs_new_protected:Nn \__enumext_redefine_item:
3162    {
3163      \RenewDocumentCommand \item { s o o }
3164        {
```

```
3165        \bool_if:nTF {##1}
3166          {
3167            \__enumext_starred_item:nn {##2} {##3}
3168          }
3169          { \__enumext_default_item:n {##2} }
3170      }
3171  }
```

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

When *tagged* PDF is active \makelabel is redefined as \hss #1 and the only way to get the align key to work correctly is by using \makebox. The solution here is to redefine \makelabel conditionally using \IfDocumentMetadataTF.

__enumext_make_label:
__enumext_make_label_std:
__enumext_make_label_box:

The function __enumext_make_label: redefine \makelabel for the keys align, font, wrap-label, wrap-label* and \item* for enumext environment. This function are passed to __enumext_list_arg_-two_X: used in the definition of the enumext environment (§13.38).

```
3172  \cs_new_protected:Nn \__enumext_make_label:
3173    {
3174      \IfDocumentMetadataTF
3175        {
3176          \__enumext_make_label_box:
3177        }
3178        {
3179          \bool_if:NTF \l__enumext_mode_box_bool
3180            {
3181              \__enumext_make_label_box:
3182            }
3183            {
3184              \__enumext_make_label_std:
3185            }
3186        }
3187    }
```

Standard definition when \DocumentMetadata is not active.

```
3188  \cs_new_protected:Nn \__enumext_make_label_std:
3189    {
3190      \RenewDocumentCommand \makelabel { m }
3191        {
3192          \tl_use:c { l__enumext_label_fill_left_ \__enumext_level: _tl }
3193          \tl_use:c { l__enumext_label_font_style_ \__enumext_level: _tl }
3194          \bool_if:cTF { l__enumext_wrap_label_ \__enumext_level: _bool }
3195            {
3196              \__enumext_item_star_exec:
3197              \use:c { __enumext_wrapper_label_ \__enumext_level: :n } { ##1 }
3198            }
3199            { ##1 }
3200          \tl_use:c { l__enumext_label_fill_right_ \__enumext_level: _tl }
3201          \tl_gclear:N \g__enumext_item_symbol_aux_tl
3202        }
3203    }
```

Definition using \makebox when \DocumentMetadata is active.

```
3204  \cs_new_protected:Nn \__enumext_make_label_box:
3205    {
3206      \RenewDocumentCommand \makelabel { m }
3207        {
3208          \makebox
3209            [ \dim_use:c { l__enumext_labelwidth_ \__enumext_level: _dim } ]
3210            [ \str_use:c { l__enumext_align_label_pos_ \__enumext_level: _str } ]
3211            {
3212              \tl_use:c { l__enumext_label_font_style_ \__enumext_level: _tl }
3213              \bool_if:cTF { l__enumext_wrap_label_ \__enumext_level: _bool }
3214                {
3215                  \__enumext_item_star_exec:
3216                  \use:c { __enumext_wrapper_label_ \__enumext_level: :n } { ##1 }
3217                }
3218                { ##1 }
3219              \tl_gclear:N \g__enumext_item_symbol_aux_tl
3220            }
3221        }
3222    }
```

(*End of definition for* __enumext_make_label:, __enumext_make_label_std:, *and* __enumext_make_label_box:.)

13.34 Setting `item-sym*` and `item-pos*` keys

In order to have a cleaner implementation of \item* for the enumext and enumext* environments it is best to define a couple of keys that allow us to control and set by default the ⟨*symbol*⟩ and its ⟨*offset*⟩.

item-sym*
item-pos*

Define and set `item-sym*` and `item-pos*` keys for enumext and enumext*.

```
3223 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
3224   {
3225     \keys_define:nn { enumext / #1 }
3226       {
3227         item-sym* .tl_set:c  = { l__enumext_item_symbol_#2_tl },
3228         item-sym* .value_required:n = true,
3229         item-sym* .initial:n  = {\textasteriskcentered},
3230         item-pos* .dim_set:c  = { l__enumext_item_symbol_sep_#2_dim },
3231         item-pos* .value_required:n = true,
3232       }
3233   }
3234 \clist_map_inline:nn
3235   {
3236     {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {enumext*}{vii}
3237   }
3238   { \__enumext_tmp:nn #1 }
```

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

13.35 Handling unknown keys

At this point in the code I already know that I will not add more ⟨*keys*⟩ and since I have already been quite *paranoid and restrictive* with the definitions of environments and commands, the only thing left to do is do it with the ⟨*keys*⟩ (you have to be consistent in life).

13.35.1 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.

```
3239 \cs_set_protected:Npn \__enumext_tmp:n #1
3240   {
3241     \keys_define:nn { enumext / #1 }
3242       {
3243         unknown .code:n = { \__enumext_keyans_unknown_keys:n {##1} }
3244       }
3245   }
3246 \clist_map_inline:nn { keyans, keyans*, keyanspic } { \__enumext_tmp:n {#1} }
```

Internal functions for handling unknown key.

```
3247 \cs_new_protected:Npn \__enumext_keyans_unknown_keys:n #1
3248   {
3249     \exp_args:NV \__enumext_keyans_unknown_keys:nn \l_keys_key_str {#1}
3250   }
3251 \cs_new_protected:Npn \__enumext_keyans_unknown_keys:nn #1#2
3252   {
3253     \tl_if_blank:nTF {#2}
3254       {
3255         \msg_error:nnn { enumext } { keyans-unknown-key } {#1}
3256       }
3257       {
3258         \msg_error:nnnn { enumext } { keyans-unknown-key-value } {#1} {#2}
3259       }
3260   }
```

(*End of definition for* unknown, __enumext_keyans_unknown_keys:n, *and* __enumext_keyans_unknown_keys:nn.)

13.35.2 Handling unknown keys for enumext*

unknown
__enumext_starred_unknown_keys:n
__enumext_starred_unknown_keys:nn

Define and set unknown key for enumext* environment.

```
3261 \keys_define:nn { enumext / enumext* }
3262   {
3263     unknown .code:n = { \__enumext_starred_unknown_keys:n {#1} }
3264   }
```

Internal functions for handling unknown key.

```
3265 \cs_new_protected:Npn \__enumext_starred_unknown_keys:n #1
3266   {
3267     \exp_args:NV \__enumext_starred_unknown_keys:nn \l_keys_key_str {#1}
3268   }
3269 \cs_new_protected:Npn \__enumext_starred_unknown_keys:nn #1#2
3270   {
3271     \tl_if_blank:nTF {#2}
3272       {
3273         \msg_error:nnn { enumext } { starred-unknown-key } {#1}
3274       }
3275       {
3276         \msg_error:nnnn { enumext } { starred-unknown-key-value } {#1} {#2}
3277       }
3278   }
```

(*End of definition for* unknown*,* __enumext_starred_unknown_keys:n*, and* __enumext_starred_unknown_keys:nn*.*)

13.35.3   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.

```
3279 \cs_set_protected:Npn \__enumext_tmp:n #1
3280   {
3281     \keys_define:nn { enumext / #1 }
3282       {
3283         unknown .code:n = { \__enumext_standar_unknown_keys:n {##1} }
3284       }
3285   }
3286 \clist_map_inline:nn { level-1,level-2,level-3,level-4 } { \__enumext_tmp:n {#1} }
```

Internal functions for handling unknown key.

```
3287 \cs_new_protected:Npn \__enumext_standar_unknown_keys:n #1
3288   {
3289     \exp_args:NV \__enumext_standar_unknown_keys:nn \l_keys_key_str {#1}
3290   }
3291 \cs_new_protected:Npn \__enumext_standar_unknown_keys:nn #1#2
3292   {
3293     \tl_if_blank:nTF {#2}
3294       {
3295         \msg_error:nnn { enumext } { standar-unknown-key } {#1}
3296       }
3297       {
3298         \msg_error:nnnn { enumext } { standar-unknown-key-value } {#1} {#2}
3299       }
3300   }
```

(*End of definition for* unknown*,* __enumext_standar_unknown_keys:n*, and* __enumext_standar_unknown_keys:nn*.*)

13.36   Redefining \item and \makelabel in keyans

The \item and \item[⟨*custom*⟩] commands work in the usual way in keyans, but the \item* and \item*[⟨*content*⟩] commands *store* the current ⟨*label*⟩ next to the ⟨*content*⟩ if it is present in the *sequence* and *prop list* defined by save-ans key.

__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.

```
3301 \cs_new_protected:Npn \__enumext_keyans_default_item:n #1
3302   {
3303     \tl_if_novalue:nTF { #1 }
3304       {
3305         \bool_set_true:N \l__enumext_wrap_label_v_bool
3306         \__enumext_item_std:w \tl_use:N \l__enumext_fake_item_indent_v_tl
3307       }
3308       {
3309         \bool_set_eq:NN \l__enumext_wrap_label_v_bool \l__enumext_wrap_label_opt_v_bool
3310         \__enumext_item_std:w [#1] \tl_use:N \l__enumext_fake_item_indent_v_tl
3311       }
3312   }
```

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

__enumext_keyans_starred_item:n

The function __enumext_keyans_starred_item:n which will make a temporary copy of the current ⟨*label*⟩, execute the show-ans or show-pos keys using the function __enumext_keyans_show_left:n and will display the ⟨*contents*⟩ of that item using the internal copy __enumext_item_std:w, this is necessary to prevent incrementing the current "*counter*" of the original ⟨*label*⟩, followed by this it will execute function __enumext_keyans_show_item_opt: handled by wrap-opt key.

```
3313 \cs_new_protected:Npn \__enumext_keyans_starred_item:n #1
3314   {
3315     \tl_set_eq:NN \l__enumext_store_current_label_tmp_tl \l__enumext_label_v_tl
3316     \__enumext_keyans_show_left:n { #1 }
3317     \bool_set_true:N \l__enumext_wrap_label_v_bool
3318     \__enumext_item_std:w \tl_use:N \l__enumext_fake_item_indent_v_tl
3319     \__enumext_keyans_show_item_opt:
```

Recover the original value of the current ⟨*label*⟩ and *store* it first in the *prop list* (including the *optional argument*), run the internal "*label and ref*" system if the save-ref key is active, *store* it in the *sequence* and finally increments \g__enumext_check_starred_cmd_int for internal check system.

```
3320     \tl_set_eq:NN \l__enumext_label_v_tl \l__enumext_store_current_label_tmp_tl
3321     \__enumext_keyans_addto_prop:n { #1 }
3322     \__enumext_keyans_store_ref:
3323     \__enumext_keyans_addto_seq:n { #1 }
3324     \int_gincr:N \g__enumext_check_starred_cmd_int
3325   }
```

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

\item*
__enumext_keyans_redefine_item:

The function __enumext_keyans_redefine_item: is responsible for adding the *starred argument* and *optional argument* by the __enumext_list_arg_two_v: function in the definition of the keyans environment. Here we need to use \peek_remove_spaces:n to prevent an unwanted space when using \item* in conjunction with the itemindent key. This function are passed to __enumext_list_arg_two_v: used in the definition of the keyans environment (§13.37.2).

```
3326 \cs_new_protected:Nn \__enumext_keyans_redefine_item:
3327   {
3328     \RenewDocumentCommand \item { s o }
3329       {
3330         \bool_if:nTF {##1}
3331           {
3332             \peek_remove_spaces:n
3333               {
3334                 \__enumext_keyans_starred_item:n {##2}
3335               }
3336           }
3337           {
3338             \__enumext_keyans_default_item:n {##2}
3339           }
3340       }
3341   }
```

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

__enumext_keyans_make_label:
__enumext_keyans_make_label_std:
__enumext_keyans_make_label_box:

The function __enumext_keyans_make_label: redefine \makelabel for the keys align, font, wrap-label, wrap-label* and \item* for keyans environment. This function are passed to __enumext_list_arg_two_v: used in the definition of the keyans environment (§13.37.2).

```
3342 \cs_new_protected:Nn \__enumext_keyans_make_label:
3343   {
3344     \IfDocumentMetadataTF
3345       {
3346         \__enumext_keyans_make_label_box:
3347       }
3348       {
3349         \bool_if:NTF \l__enumext_mode_box_bool
3350           {
3351             \__enumext_keyans_make_label_box:
3352           }
3353           {
3354             \__enumext_keyans_make_label_std:
3355           }
3356       }
3357   }
```

Standard definition when `\DocumentMetadata` is not active.

```
3358  \cs_new_protected:Nn \__enumext_keyans_make_label_std:
3359    {
3360      \RenewDocumentCommand \makelabel { m }
3361        {
3362          \tl_use:N \l__enumext_label_fill_left_v_tl
3363          \tl_use:N \l__enumext_label_font_style_v_tl
3364          \bool_if:NTF \l__enumext_wrap_label_v_bool
3365            {
3366              \__enumext_wrapper_label_v:n { ##1 }
3367            }
3368            { ##1 }
3369          \tl_use:N \l__enumext_label_fill_right_v_tl
3370        }
3371    }
```

Definition using `\makebox` when `\DocumentMetadata` is active.

```
3372  \cs_new_protected:Nn \__enumext_keyans_make_label_box:
3373    {
3374      \RenewDocumentCommand \makelabel { m }
3375        {
3376          \makebox[ \l__enumext_labelwidth_v_dim ][ \l__enumext_align_label_pos_v_str ]
3377            {
3378              \tl_use:N \l__enumext_label_font_style_v_tl
3379              \bool_if:NTF \l__enumext_wrap_label_v_bool
3380                {
3381                  \__enumext_wrapper_label_v:n { ##1 }
3382                }
3383                { ##1 }
3384            }
3385        }
3386    }
```

(*End of definition for* `\__enumext_keyans_make_label:`, `\__enumext_keyans_make_label_std:`, *and* `\__enumext_keyans_-make_label_box:`.)

13.37 Second argument of the lists

At this point of the code we have already programmed most the necessary tools to create a custom `list` environment, remember that the function `\__enumext_start_list:nn` takes two arguments, the first one we have ready, the second one we will define for all the levels of the environment `enumext` and the environment `keyans`.

13.37.1 Calculation of `\leftmargin` and `\itemindent`

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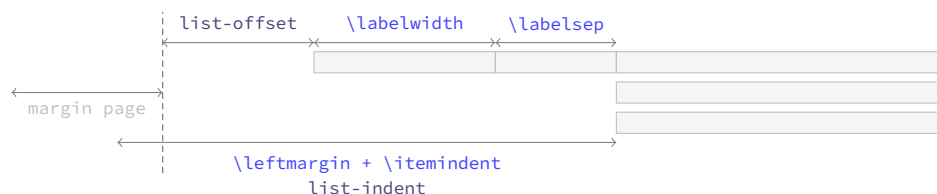
Figure 9: Representation of standard horizontal lengths in `list` environment.

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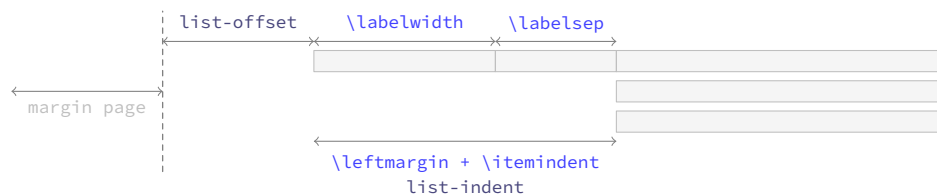
Figure 10: Representation of horizontal lengths concept in list in enumext.

Where the default values will look like in the figure 11.

`\__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:
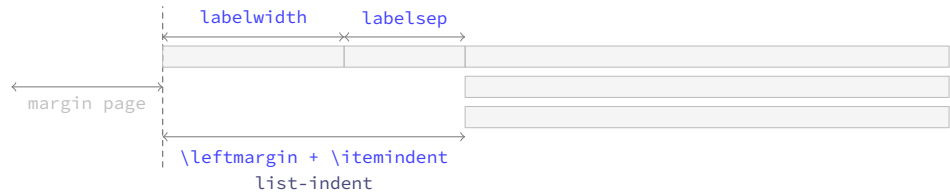
Figure 11: Default horizontal lengths in enumext.

```
#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 (§13.37.2).

```
3387  \cs_new_protected:Npn \__enumext_calc_hspace:NNNNNNN #1 #2 #3 #4 #5 #6 #7
3388    {
3389      \dim_compare:nNnT { #1 } < { \c_zero_dim }
3390        {
3391          \msg_warning:nnnV { enumext } { width-non-positive }{ labelwidth }{ #1 }
3392          \dim_set:Nn #1 { \dim_abs:n { #1 } }
3393        }
3394      \dim_compare:nNnT { #2 } < { \c_zero_dim }
3395        {
3396          \msg_warning:nnnV { enumext } { width-negative }{ labelsep }{ #2 }
3397          \dim_set:Nn #2 { \dim_abs:n { #2 } }
3398        }
```

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

```
3399        \bool_if:nF #7 { \dim_set:Nn #4 { #1 + #2} }
```

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

```
3400        \dim_compare:nNnTF { #4 } < { \c_zero_dim }
3401          {
3402          \dim_set:Nn #6 { #1 + #2 - #4}
3403          \dim_set:Nn #5 { #1 + #2 + #3 - #6 }
3404          }
3405          {
3406          \dim_compare:nNnT { #4 } = { #1 + #2 }
3407            { \dim_set:Nn #6 { \c_zero_dim } }
3408          \dim_compare:nNnT { #4 } < { #1 + #2 }
3409            { \dim_set:Nn #6 { #1 + #2 - #4} }
3410          \dim_compare:nNnT { #4 } > { #1 + #2 }
3411            {
3412            \dim_set:Nn #6 { -#1 - #2 + #4}
3413            \dim_set:Nn #6 { #6*-1}
3414            }
3415          \dim_set:Nn #5 { #1 + #2 + #3 - #6 }
3416          }
3417      }
3418  \cs_generate_variant:Nn \__enumext_calc_hspace:NNNNNNN { ccccccc }
```

(End of definition for `\__enumext_calc_hspace:NNNNNNN`.)

13.37.2 Setting second argument of the lists

`\__enumext_list_arg_two_i:`
`\__enumext_list_arg_two_ii:`
`\__enumext_list_arg_two_iii:`
`\__enumext_list_arg_two_iv:`
`\__enumext_list_arg_two_v:`

We will "not set" `\leftmargini`, `\leftmarginii`, `\leftmarginiii` or `\leftmarginiv`, in this case, we will directly set the parameters for vertical and horizontal list spacing per level.

```
3419  \cs_set_protected:Npn \__enumext_tmp:n #1
3420    {
3421      \cs_new_protected:cpn { __enumext_list_arg_two_#1: }
3422        {
3423          \__enumext_calc_hspace:ccccccc
3424            { l__enumext_labelwidth_#1_dim } { l__enumext_labelsep_#1_dim }
3425            { l__enumext_listoffset_#1_dim } { l__enumext_leftmargin_tmp_#1_dim }
3426            { l__enumext_leftmargin_#1_dim } { l__enumext_itemindent_#1_dim }
3427            { l__enumext_leftmargin_tmp_#1_bool }
```

```
3428            \clist_map_inline:nn
3429              { labelsep, labelwidth, itemindent, leftmargin, rightmargin, listparindent }
3430              { \dim_set_eq:cc {####1} { l__enumext_####1_#1_dim } }
3431            \clist_map_inline:nn { topsep, parsep, partopsep, itemsep }
3432              { \skip_set_eq:cc {####1} { l__enumext_####1_#1_skip } }
3433            \usecounter { enumX#1 }
3434            \setcounter { enumX#1 } { \int_eval:n { \int_use:c { l__enumext_start_#1_int } - 1 } }
3435            \str_if_eq:nnTF {#1} { v }
3436              {
3437                \__enumext_keyans_redefine_item:
3438                \__enumext_keyans_make_label:
3439                \__enumext_keyans_ref:
3440                \__enumext_keyans_fake_item_indent:
3441                \bool_if:cT { l__enumext_show_length_#1_bool }
3442                  {
3443                    \msg_term:nnnn { enumext } { list-lengths-not-nested } { v } { keyans }
3444                  }
3445              }
3446              {
3447                \__enumext_redefine_item:
3448                \__enumext_make_label:
3449                \__enumext_standar_ref:
3450                \__enumext_fake_item_indent:
3451                \bool_if:cT { l__enumext_show_length_#1_bool }
3452                  {
3453                    \msg_term:nnne { enumext } { list-lengths } {#1}
3454                      { \int_use:N \l__enumext_level_int }
3455                  }
3456              }
3457          }
3458      }
3459    \clist_map_inline:nn { i, ii, iii, iv, v } { \__enumext_tmp:n {#1} }
```

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

__enumext_list_arg_two_vii:
__enumext_list_arg_two_viii:

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

```
3460    \cs_set_protected:Npn \__enumext_tmp:n #1
3461      {
3462        \cs_new_protected:cpn { __enumext_list_arg_two_#1: }
3463          {
3464            \bool_set_true:c { l__enumext_leftmargin_tmp_#1_bool }
3465            \dim_zero:c { l__enumext_leftmargin_tmp_#1_dim }
3466            \__enumext_calc_hspace:ccccccc
3467              { l__enumext_labelwidth_#1_dim } { l__enumext_labelsep_#1_dim }
3468              { l__enumext_listoffset_#1_dim } { l__enumext_leftmargin_tmp_#1_dim }
3469              { l__enumext_leftmargin_#1_dim } { l__enumext_itemindent_#1_dim }
3470              { l__enumext_leftmargin_tmp_#1_bool }
3471            \clist_map_inline:nn
3472              { labelsep, labelwidth, itemindent, leftmargin, rightmargin, listparindent }
3473              { \dim_set_eq:cc {####1} { l__enumext_####1_#1_dim } }
3474            \clist_map_inline:nn { topsep, parsep, partopsep, itemsep }
3475              { \skip_set_eq:cc {####1} { l__enumext_####1_#1_skip } }
3476            \skip_set_eq:Nc \parsep  { l__enumext_itemsep_#1_skip }
3477            \skip_zero:N \partopsep
3478            \usecounter { enumX#1 }
3479            \setcounter { enumX#1 } { \int_eval:n { \int_use:c { l__enumext_start_#1_int } - 1 } }
3480            \__enumext_starred_ref:
3481            \str_if_eq:nnTF {#1} { vii }
3482              {
3483                \__enumext_fake_item_indent_vii:
3484                \bool_if:cT { l__enumext_show_length_vii_bool }
3485                  { \msg_term:nnnn { enumext } { list-lengths-not-nested } { vii } { enumext* } }
3486              }
3487              {
3488                \__enumext_fake_item_indent_viii:
3489                \bool_if:cT { l__enumext_show_length_#1_bool }
3490                  { \msg_term:nnnn { enumext } { list-lengths-not-nested } { #1 } { keyans* } }
3491              }
```

```
3492          }
3493     }
3494   \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:.)

13.38   The environment enumext

__enumext_safe_exec:

The __enumext_safe_exec: function first call the function __enumext_internal_mini_page: to create the environment __enumext_mini_page, then the function __enumext_is_not_nested: which sets \g__enumext_standar_bool to *"true"* if we are not nested within enumext*, we will increment \l__enumext_level_int to restrict nesting of the environment, set \l__enumext_standar_bool to *"true"* and finally call the function __enumext_is_on_first_level: which sets \l__enumext_standar_first_bool to *"true"* only if the environment is not nested and we are at the *"first level"*.

```
3495   \cs_new_protected:Nn \__enumext_safe_exec:
3496     {
3497       \__enumext_internal_mini_page:
3498       \__enumext_is_not_nested:
3499       \int_incr:N \l__enumext_level_int
3500       \int_compare:nNnT { \l__enumext_level_int } > { 4 }
3501         { \msg_fatal:nn { enumext } { list-too-deep } }
3502       \bool_set_true:N \l__enumext_standar_bool
3503       \bool_set_false:N \l__enumext_starred_bool
3504       \__enumext_is_on_first_level:
3505     }
```

(*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.

```
3506   \cs_new_protected:Npn \__enumext_parse_keys:n #1
3507     {
3508       \tl_if_novalue:nF {#1}
3509         {
3510           \str_clear:N \l__enumext_series_str
3511           \int_compare:nNnTF { \l__enumext_level_int } = { 1 }
3512             {
3513               \keys_set:nn { enumext / level-1 } {#1}
3514               \__enumext_parse_series:n {#1}
3515               \__enumext_nested_base_line_fix:
3516             }
3517             {
3518               \exp_args:Ne \keys_set:nn
3519                 { enumext / level-\int_use:N \l__enumext_level_int } {#1}
3520             }
3521           \__enumext_store_active_keys:n {#1}
3522         }
3523     }
```

(*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*.

```
3524   \cs_new_protected:Nn \__enumext_start_store_level:
3525     {
3526       \bool_lazy_all:nT
3527         {
3528           { \bool_if_p:N \l__enumext_store_active_bool }
3529           { \bool_not_p:n { \l__enumext_keyans_env_bool } }
3530           { \bool_if_p:N \g__enumext_standar_bool }
3531         }
3532         {
3533           \int_compare:nNnT { \l__enumext_level_int } > { 1 }
3534             {
3535               \bool_set_true:c { l__enumext_store_upper_level_ \__enumext_level: _bool }
3536               \__enumext_store_level_open:
3537             }
3538         }
```

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

```
3539      \bool_lazy_all:nT
3540        {
3541          { \bool_if_p:N \l__enumext_store_active_bool }
3542          { \bool_not_p:n { \l__enumext_keyans_env_bool } }
3543          { \int_compare_p:nNn { \l__enumext_level_h_int } = { 1 } }
3544        }
3545        {
3546          \int_compare:nNnT { \l__enumext_level_int } > { 0 }
3547            {
3548              \bool_set_true:c { l__enumext_store_upper_level_ \__enumext_level: _bool }
3549              \__enumext_store_level_open:
3550            }
3551        }
3552      }
```

(*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*`.

```
3553  \cs_new_protected:Nn \__enumext_stop_store_level:
3554    {
3555      \bool_if:cT { l__enumext_store_upper_level_ \__enumext_level: _bool }
3556        {
3557          \__enumext_store_level_close:
3558        }
3559    }
```

(*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.

```
3560  \cs_new_protected:Nn \__enumext_multicols_start:
3561    {
3562      \int_compare:nNnT
3563        { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } > { 1 }
3564        {
3565          \dim_compare:nNnT
3566            { \dim_use:c { l__enumext_columns_sep_ \__enumext_level: _dim } } = { \c_zero_dim }
3567            {
3568              \dim_set:cn { l__enumext_columns_sep_ \__enumext_level: _dim }
3569                {
3570                  ( \dim_use:c { l__enumext_labelwidth_ \__enumext_level: _dim }
3571                    + \dim_use:c { l__enumext_labelsep_ \__enumext_level: _dim }
3572                  ) / \int_use:c { l__enumext_columns_ \__enumext_level: _int }
3573                    - \dim_use:c { l__enumext_listoffset_ \__enumext_level: _dim }
3574                }
3575            }
3576          \dim_set_eq:Nc \columnsep { l__enumext_columns_sep_ \__enumext_level: _dim  }
3577          \int_compare:nNnT { \l__enumext_level_int } > { 1 }
3578            {
3579              \dim_zero:N \columnseprule
3580            }
```

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.

```
3581          \bool_if:cF { l__enumext_minipage_active_ \__enumext_level: _bool }
3582            {
3583              \skip_zero:N \multicolsep
3584              \__enumext_multi_addvspace:
3585            }
3586          \raggedcolumns
3587          \begin{multicols}{ \int_use:c { l__enumext_columns_ \__enumext_level: _int } }
3588        }
3589    }
```

(*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:.

```
3590  \cs_new_protected:Nn \__enumext_multicols_stop:
3591    {
3592      \int_compare:nNnTF
3593        { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } > { 1 }
3594        {
3595          \__enumext_stop_list:
3596          \__enumext_stop_store_level:
3597          \end{multicols}
3598          \__enumext_unskip_unkern:
3599          \__enumext_unskip_unkern:
3600          \par\addvspace{ \skip_use:c { l__enumext_multicols_below_ \__enumext_level: _skip } }
3601        }
3602        {
3603          \__enumext_stop_list:
3604          \__enumext_stop_store_level:
3605        }
3606    }
```

(*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.

```
3607  \cs_new_protected:Nn \__enumext_before_list:
3608    {
3609      \__enumext_vspace_above:
3610      \__enumext_before_args_exec:
3611      \__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.

```
3612      \dim_compare:nNnT
3613        { \dim_use:c { l__enumext_minipage_right_ \__enumext_level: _dim } } > { \c_zero_dim }
3614        {
3615          \dim_set:cn { l__enumext_minipage_left_ \__enumext_level: _dim }
3616            {
3617              \linewidth
3618              - \dim_use:c { l__enumext_minipage_right_ \__enumext_level: _dim }
3619              - \dim_use:c { l__enumext_minipage_hsep_ \__enumext_level: _dim }
3620            }
```

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.

```
3621          \bool_set_true:c { l__enumext_minipage_active_ \__enumext_level: _bool }
3622          \int_gincr:N \g__enumext_minipage_stat_int
3623          \__enumext_minipage_add_space:
3624          \noindent
3625          \__enumext_mini_page{ \dim_use:c { l__enumext_minipage_left_ \__enumext_level: _dim } }
3626        }
3627      \__enumext_multicols_start:
3628    }
```

(*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.

```
3629  \cs_new_protected:Nn \__enumext_second_part:
```

```
3630      {
3631        \bool_if:cTF { l__enumext_minipage_active_ \__enumext_level: _bool }
3632          {
3633            \int_compare:nNnT { \g__enumext_minipage_stat_int } = { 1 }
3634              {
3635                \msg_warning:nn { enumext } { missing-miniright }
3636                \miniright
3637              }
3638            \int_gzero:N \g__enumext_minipage_stat_int
3639            \__enumext_unskip_unkern: % remove topsep + [partopsep]
3640            \end__enumext_mini_page
3641          }
3642          {
3643            \__enumext_multicols_stop:
3644          }
```

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.

```
3645      \__enumext_after_stop_list:
3646      \__enumext_check_ans_key_hook:
3647      \__enumext_vspace_below:
3648      \bool_set_false:N \l__enumext_standar_bool
3649      \__enumext_resume_save_counter:
3650    }
```

(*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.

```
3651 \cs_new_protected:Nn \__enumext_set_item_width:
3652   {
3653     \dim_set:Nn \itemwidth { \linewidth }
3654     \dim_compare:nT
3655       {
3656         \dim_use:c { l__enumext_listoffset_ \__enumext_level: _dim } != \c_zero_dim
3657       }
3658       {
3659         \dim_sub:Nn \itemwidth
3660           {
3661             \dim_use:c { l__enumext_listoffset_ \__enumext_level: _dim }
3662           }
3663       }
3664   }
```

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

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

```
3665 \NewDocumentEnvironment{enumext}{ O{} }
3666   {
3667     \__enumext_safe_exec:
3668     \__enumext_parse_keys:n {#1}
3669     \__enumext_before_list:
3670     \__enumext_start_store_level:
3671     \__enumext_start_list:nn
3672       { \tl_use:c { l__enumext_label_ \__enumext_level: _tl } }
3673       {
3674         \use:c { __enumext_list_arg_two_ \__enumext_level: : }
3675         \__enumext_before_keys_exec:
3676       }
3677     \__enumext_set_item_width:
3678     \__enumext_after_args_exec:
3679   }
3680   {
3681     \__enumext_second_part:
3682   }
```

(*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.

```
3683  \__enumext_after_env:nn {enumext}
3684    {
3685      \__enumext_execute_after_env:
3686    }
```

13.39   The environment keyans

The environment keyans also based on lists. The main differences with the enumext environment are the *nesting* and the way the *answers* (choice) will be stored and checked, this environment is intended exclusively for *"multiple choice questions"*.

__enumext_keyans_safe_exec: The keyans environment will only be available if the save-ans key is active and can only be used at the *"first level"* within the enumext environment. We do not want the environment to be nested, so we will set a maximum at this point. If the conditions are not met, an error message will be returned.

```
3687  \cs_new_protected:Nn \__enumext_keyans_safe_exec:
3688    {
3689      \bool_if:NF \l__enumext_store_active_bool
3690        {
3691          \msg_error:nnnn { enumext } { wrong-place }{ keyans }{ save-ans }
3692        }
3693      \int_incr:N \l__enumext_keyans_level_int
3694      \bool_set_true:N \l__enumext_keyans_env_bool
3695      \__enumext_keyans_name_and_start:
3696      % Set false for interfering with enumext nested in keyans (yes, its possible and crayze)
3697      \bool_set_false:N \l__enumext_store_active_bool
3698      \int_compare:nNnT { \l__enumext_keyans_level_int } > { 1 }
3699        {
3700          \msg_error:nn { enumext } { keyans-nested }
3701        }
3702      \int_compare:nNnT { \l__enumext_level_int } > { 1 }
3703        {
3704          \msg_error:nn { enumext } { keyans-wrong-level }
3705        }
3706    }
```

(*End of definition for* __enumext_keyans_safe_exec:.)

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

```
3707  \cs_new_protected:Npn \__enumext_keyans_parse_keys:n #1
3708    {
3709      \keys_set:nn { enumext / keyans } {#1}
3710    }
```

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

__enumext_before_list_v:
__enumext_keyans_multicols_start:
__enumext_keyans_multicols_stop:
__enumext_second_part_v:

Same implementation as the one used in the enumext environment.

```
3711  \cs_new_protected:Nn \__enumext_before_list_v:
3712    {
3713      \__enumext_vspace_above_v:
3714      \__enumext_before_args_exec_v:
3715      \dim_compare:nNnT { \l__enumext_minipage_right_v_dim } > { \c_zero_dim }
3716        {
3717          \dim_set:Nn \l__enumext_minipage_left_v_dim
3718            {
3719              \linewidth - \l__enumext_minipage_right_v_dim - \l__enumext_minipage_hsep_v_dim
3720            }
3721          \bool_set_true:N \l__enumext_minipage_active_v_bool
3722          \int_gincr:N \g__enumext_minipage_stat_int
3723          \__enumext_keyans_minipage_add_space:
3724          \__enumext_mini_page{ \l__enumext_minipage_left_v_dim }
3725        }
3726      \__enumext_keyans_multicols_start:
3727    }
3728  \cs_new_protected:Nn \__enumext_keyans_multicols_start:
3729    {
3730      \int_compare:nNnT { \l__enumext_columns_v_int } > { 1 }
3731        {
3732          \dim_compare:nNnT { \l__enumext_columns_sep_v_dim } = { \c_zero_dim }
3733            {
3734              \dim_set:Nn \l__enumext_columns_sep_v_dim
3735                {
```

```
3736                    (
3737                        \l__enumext_labelwidth_v_dim + \l__enumext_labelsep_v_dim
3738                    ) / \l__enumext_columns_v_int
3739                   - \l__enumext_listoffset_v_dim
3740                }
3741            }
3742        \dim_set_eq:NN \columnsep \l__enumext_columns_sep_v_dim
3743        \dim_zero:N \columnseprule % no rule here
3744        \bool_if:NF \l__enumext_minipage_active_v_bool
3745            {
3746                \skip_zero:N \multicolsep
3747                \__enumext_keyans_multi_addvspace:
3748            }
3749        \raggedcolumns
3750        \begin{multicols}{ \l__enumext_columns_v_int }
3751        }
3752    }
3753 \cs_new_protected:Nn \__enumext_keyans_multicols_stop:
3754    {
3755        \int_compare:nNnTF { \l__enumext_columns_v_int } > { 1 }
3756            {
3757                \__enumext_stop_list:
3758                \end{multicols}
3759                \__enumext_unskip_unkern:
3760                \__enumext_unskip_unkern:
3761                \par\addvspace{ \l__enumext_multicols_below_v_skip }
3762            }
3763            {
3764                \__enumext_stop_list:
3765            }
3766    }
3767 \cs_new_protected:Nn \__enumext_second_part_v:
3768    {
3769        \bool_if:NTF \l__enumext_minipage_active_v_bool
3770            {
3771                \int_compare:nNnT { \g__enumext_minipage_stat_int } = { 1 }
3772                    {
3773                        \msg_warning:nn { enumext } { missing-miniright }
3774                        \miniright
3775                    }
3776                \int_gzero:N \g__enumext_minipage_stat_int
3777                \__enumext_unskip_unkern: % remove \topsep + [\partopsep]
3778                \end__enumext_mini_page
3779                \par\addvspace{ \l__enumext_minipage_after_skip }
3780            }
3781            {
3782                \__enumext_keyans_multicols_stop:
3783            }
3784        \bool_set_false:N \l__enumext_keyans_env_bool
3785        \__enumext_after_stop_list_v:
3786        \__enumext_vspace_below_v:
3787    }
```

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

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

```
3788 \cs_new_protected:Nn \__enumext_keyans_set_item_width:
3789    {
3790        \dim_set:Nn \itemwidth { \linewidth }
3791        \dim_compare:nT
3792            {
3793                \l__enumext_listoffset_v_dim != \c_zero_dim
3794            }
3795            {
3796                \dim_sub:Nn \itemwidth { \l__enumext_listoffset_v_dim }
3797            }
3798    }
```

(*End of definition for* __enumext_keyans_set_item_width:.)

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

```
3799  \NewDocumentEnvironment{keyans}{ O{} }
3800    {
3801      \__enumext_keyans_safe_exec:
3802      \__enumext_keyans_parse_keys:n {#1}
3803      \__enumext_before_list_v:
3804      \__enumext_start_list:nn
3805        { \tl_use:N \l__enumext_label_v_tl }
3806        {
3807          \__enumext_list_arg_two_v:
3808          \__enumext_before_keys_exec_v:
3809        }
3810      \__enumext_keyans_set_item_width:
3811      \__enumext_after_args_exec_v:
3812    }
3813    {
3814      \__enumext_check_starred_cmd:n { item }
3815      \__enumext_second_part_v:
3816    }
```

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

13.40   Tagging PDF support for non-standart list environments

The LaTeX release 2022-06-01 brings automatic support for *tagged* PDF in several aspects, including the standard *list environments* and the list environment. Unfortunately non-standard *list environments* like keyanspic or the horizontal list environments enumext* and keyans* are not structured in a nice way, i.e. the expected result in the PDF file is the expected one, but the underlying structure is not correct. In simple terms, for *tagged* PDF a list environment is a list environment, no matter what it looks like in the PDF file.

To maintain a correct list structure when \DocumentMetadata is active, it is necessary to do some things manually. This implementation is an adaptation of my answer thanks to Ulrike Fischer's comments in How can I modify my \item redefinition to be compatible with tagging-pdf.

13.40.1   Socket for tagging support in enumext* and keyans*

start-list-tags   We will first define the necessary sockets and their behavior for enumext* and keyans*.
stop-start-tags
stop-list-tags
__enumext_start_list_tag:n
__enumext_stop_start_list_tag:
__enumext_stop_list_tag:n

```
3817  \socket_new:nn {tagsupport/enumext/starred}{ 1 }
3818  \socket_new_plug:nnn {tagsupport/enumext/starred} {start-list-tags}
3819    {
3820      \tag_resume:n {#1}
3821      \tag_struct_begin:n {tag=LI}
3822      \tag_struct_begin:n {tag=Lbl}
3823      \tag_mc_begin:n {tag=Lbl}
3824    }
3825  \socket_new_plug:nnn {tagsupport/enumext/starred} {stop-start-tags}
3826    {
3827      \tag_mc_end:
3828      \tag_struct_end:n {tag=Lbl}
3829      \tag_struct_begin:n {tag=LBody}
3830      \tag_struct_begin:n {tag=text-unit}
3831      \tag_struct_begin:n {tag=text}
3832    }
3833  \socket_new_plug:nnn {tagsupport/enumext/starred} {stop-list-tags}
3834    {
3835      \tag_struct_end:n {tag=text}
3836      \tag_struct_end:n {tag=text-unit}
3837      \tag_struct_end:n {tag=LBody}
3838      \tag_struct_end:n {tag=LI}
3839      \tag_suspend:n {#1}
3840    }
```

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

```
3841  \cs_new_protected_nopar:Npn \__enumext_start_list_tag:n #1
3842    {
3843      \IfDocumentMetadataTF
3844        {
3845          \socket_assign_plug:nn {tagsupport/enumext/starred} {start-list-tags}
3846          \socket_use:n {tagsupport/enumext/starred} {#1}
3847        } {}
3848    }
3849  \cs_new_protected_nopar:Nn \__enumext_stop_start_list_tag:
3850    {
```

```
3851        \IfDocumentMetadataTF
3852          {
3853            \socket_assign_plug:nn {tagsupport/enumext/starred} {stop-start-tags}
3854            \socket_use:nn {tagsupport/enumext/starred} { }
3855          } {}
3856      }
3857  \cs_new_protected_nopar:Npn \__enumext_stop_list_tag:n #1
3858    {
3859      \IfDocumentMetadataTF
3860        {
3861          \socket_assign_plug:nn {tagsupport/enumext/starred} {stop-list-tags}
3862          \socket_use:nn {tagsupport/enumext/starred} {#1}
3863        } {}
3864    }
```

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

13.40.2 Socket for tagging support in keyanspic

start-list-tags
stop-start-tags
stop-list-tags
__enumext_anspic_start_list_tag:
__enumext_anspic_stop_start_list_tag:
__enumext_anspic_stop_list_tag:

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

```
3865  \socket_new:nn {tagsupport/enumext/keyanspic}{ 0 }
3866  \socket_new_plug:nnn {tagsupport/enumext/keyanspic} {start-list-tags}
3867    {
3868      \tag_resume:n {keyanspic}
3869      \tag_struct_begin:n {tag=LI}
3870      \tag_struct_begin:n {tag=Lbl}
3871      \tag_mc_begin:n {tag=Lbl}
3872    }
3873  \socket_new_plug:nnn {tagsupport/enumext/keyanspic} {stop-start-tags}
3874    {
3875      \tag_mc_end:
3876      \tag_struct_end:n {tag=Lbl}
3877      \tag_struct_begin:n {tag=LBody}
3878      \tag_struct_begin:n {tag=text-unit}
3879      \tag_struct_begin:n {tag=text}
3880      \tag_mc_begin:n {tag=text}
3881    }
3882  \socket_new_plug:nnn {tagsupport/enumext/keyanspic} {stop-list-tags}
3883    {
3884      \tag_mc_end:
3885      \tag_struct_end:n {tag=text-unit}
3886      \tag_struct_end:n {tag=text}
3887      \tag_struct_end:n {tag=LBody}
3888      \tag_struct_end:n {tag=LI}
3889      \tag_suspend:n {keyanspic}
3890    }
```

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

```
3891  \cs_new_protected_nopar:Nn \__enumext_anspic_start_list_tag:
3892    {
3893      \IfDocumentMetadataTF
3894        {
3895          \socket_assign_plug:nn {tagsupport/enumext/keyanspic} {start-list-tags}
3896          \socket_use:n {tagsupport/enumext/keyanspic}
3897        } {}
3898    }
3899  \cs_new_protected_nopar:Nn \__enumext_anspic_stop_start_list_tag:
3900    {
3901      \IfDocumentMetadataTF
3902        {
3903          \socket_assign_plug:nn {tagsupport/enumext/keyanspic} {stop-start-tags}
3904          \socket_use:nn {tagsupport/enumext/keyanspic}
3905        } {}
3906    }
3907  \cs_new_protected_nopar:Nn \__enumext_anspic_stop_list_tag:
3908    {
3909      \IfDocumentMetadataTF
3910        {
3911          \socket_assign_plug:nn {tagsupport/enumext/keyanspic} {stop-list-tags}
3912          \socket_use:nn {tagsupport/enumext/keyanspic}
3913        } {}
3914    }
```

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

13.41 The environment `keyanspic` and `\anspic`

The `keyanspic` environment is a `list` based environment that uses the same configuration for *"spacing"* and ⟨*label*⟩ as the `keyans` environment, but it does not use `\item`. The ⟨*contents*⟩ are passed to the environment by means of the `\anspic` command as replacement for `\item` command and placed inside `minipage` environments, with the ⟨*label*⟩ centered *"above"* or *"below"*, adjusting *widths* and *position* according to the options passed to the environment.
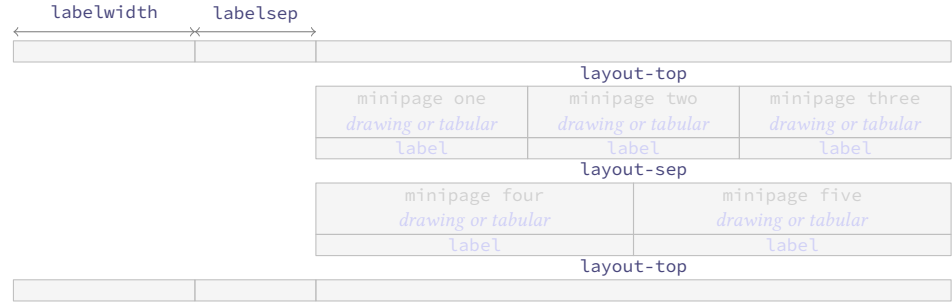
Figure 12: Representation of the `keyanspic` spacing in enumext.

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?.

13.41.1 The environment `keyanspic`

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`.

```
3915 \keys_define:nn { enumext / keyanspic }
3916   {
3917     label-pos .choice:,
3918     label-pos / above   .code:n =
3919                          \bool_set_true:N \l__enumext_anspic_label_above_bool
3920                          \str_set:Nn \l__enumext_anspic_mini_pos_str { t },
3921     label-pos / below   .code:n =
3922                          \bool_set_false:N \l__enumext_anspic_label_above_bool
3923                          \str_set:Nn \l__enumext_anspic_mini_pos_str { b },
3924     label-pos / unknown .code:n =
3925                          \msg_error:nneee { enumext } { unknown-choice }
3926                            { label-pos } { above,~ below } { \exp_not:n {#1} },
3927     label-pos  .initial:n       = below,
3928     label-pos  .value_required:n = true,
3929     label-sep  .skip_set:N       = \l__enumext_anspic_label_sep_skip,
3930     label-sep  .value_required:n = true,
3931     layout-sty .tl_set:N         = \l__enumext_anspic_layout_style_tl,
3932     layout-sty .value_required:n = true,
3933     layout-sep .code:n           = \keys_set:nn { enumext / keyans }
3934                                      { parsep = #1 },
3935     layout-sep .value_required:n = true,
3936     layout-top .code:n           = \keys_set:nn { enumext / keyans }
3937                                      { topsep = #1 },
3938     layout-top .value_required:n = true,
3939     unknown    .code:n           = { \__enumext_keyans_unknown_keys:n {#1} }
3940   }
```

(*End of definition for* `label-pos` *and others.*)

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

```
3941 \cs_new_protected:Nn \__enumext_keyans_pic_safe_exec:
3942   {
3943     \int_incr:N \l__enumext_keyans_pic_level_int
3944     \int_compare:nNnT { \l__enumext_keyans_pic_level_int } > { 1 }
3945       {
```

```
3946              \msg_error:nn { enumext } { keyanspic-nested }
3947            }
3948        \__enumext_keyans_name_and_start:
3949      }
```

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

```
3950 \cs_new_protected:Npn \__enumext_keyans_pic_parse_keys:n #1
3951    {
3952      \tl_if_novalue:nF {#1}
3953        {
3954          \keys_set:nn { enumext / keyanspic } {#1}
3955        }
3956    }
```

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

```
3957 \cs_new_protected:Npn \__enumext_keyans_pic_skip_abs:N #1
3958    {
3959      \dim_compare:nNnT { #1 } < { \c_zero_dim }
3960        {
3961          \skip_set:Nn #1 { -#1 }
3962        }
3963    }
```

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.

```
3964 \cs_new_protected:Npn \__enumext_keyans_pic_arg_two:
3965    {
3966      \bool_set_false:N \l__enumext_leftmargin_tmp_v_bool
3967      \__enumext_list_arg_two_v:
3968      \__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*.

```
3969        \bool_if:NF \l__enumext_anspic_label_above_bool
3970          {
3971            \stepcounter { enumXv }
3972            \hbox_set:Nn \l__enumext_anspic_label_box { \l__enumext_label_v_tl }
3973            \dim_set:Nn \l__enumext_anspic_label_htdp_dim
3974              {
3975                \box_ht_plus_dp:N \l__enumext_anspic_label_box
3976              }
3977            \skip_add:Nn \parsep
3978              {
3979                \l__enumext_anspic_label_htdp_dim
3980                + \box_dp:N \strutbox
3981                + \l__enumext_anspic_label_sep_skip
3982              }
3983          }
```

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.

```
3984      \dim_add:Nn  \leftmargin { -\labelwidth - \labelsep }
3985      \skip_add:Nn \topsep { 0.5\box_dp:N \strutbox }
3986      \dim_zero:N  \listparindent
3987      \skip_zero:N \partopsep
3988      \skip_zero:N \itemsep
3989    }
```

(*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`.

```
3990 \NewDocumentEnvironment{keyanspic}{ o }
3991    {
3992      \__enumext_keyans_pic_safe_exec:
3993      \__enumext_keyans_pic_parse_keys:n {#1}
```

```
3994      \begin{list} { } { \__enumext_keyans_pic_arg_two: }
3995      \IfDocumentMetadataTF
3996        {
3997          \tag_suspend:n {list}
3998        }{}
3999      \item[] \scan_stop:
4000      % paranoia
4001      \RenewDocumentCommand \item {}
4002        {
4003          \msg_error:nn { enumext } { keyanspic-item-cmd }
4004        }
4005      \IfDocumentMetadataTF
4006        {
4007          \tag_resume:n {keyanspic}
4008          \tag_tool:n {para/tagging=false}
4009          \tag_suspend:n {keyanspic}
4010        } { }
4011    }
4012    {
4013      \IfDocumentMetadataTF
4014        {
4015          \tag_resume:n {keyanspic}
4016          \tag_struct_begin:n {tag=L,attribute=enumerate}
4017        } { }
```

Now we process the command \anspic, if the key layout-sty is not present, the number of times the \anspic command appears will be counted from \l__enumext_anspic_args_seq and placed a *single line*.

```
4018      \__enumext_anspic_exec:
4019      \IfDocumentMetadataTF
4020        {
4021          \tag_suspend:n {keyanspic}
4022        } { }
4023    \end{list}
4024    \IfDocumentMetadataTF
4025      {
4026        \tag_struct_end:
4027        \tag_struct_end:
4028      } { }
```

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

```
4029      \__enumext_check_starred_cmd:n { anspic }
4030      \setcounter { enumXvi } { 0 }
4031      \bool_if:NTF \l__enumext_anspic_label_above_bool
4032        {
4033          \par\addvspace{ 0.5\box_dp:N \strutbox }
4034        }
4035        {
4036          \par
4037          \addvspace
4038            {
4039              \dim_eval:n
4040                {
4041                  \l__enumext_anspic_label_htdp_dim + \box_ht_plus_dp:N \strutbox
4042                  + \l__enumext_anspic_label_sep_skip + \l__enumext_topsep_v_skip
4043                }
4044            }
4045        }
4046    }
```

(*End of definition for* keyanspic. *This function is documented on page 15.*)

13.41.2 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.

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.

```
4047  \NewDocumentCommand \anspic { s o +m }
4048    {
4049      \bool_if:NF \l__enumext_store_active_bool
4050        {
4051          \msg_error:nnnn { enumext } { wrong-place }{ keyanspic }{ save-ans }
4052        }
4053      \int_compare:nNnT { \l__enumext_level_int } > { 1 }
4054        {
4055          \msg_error:nn { enumext } { keyanspic-wrong-level }
4056        }
4057      \int_compare:nNnT { \l__enumext_keyans_level_int } = { 1 }
4058        {
4059          \msg_error:nnnn { enumext } { command-wrong-place }{ anspic }{ keyans }
4060        }
4061      \seq_put_right:Nn \l__enumext_anspic_args_seq
4062        {
4063          \__enumext_anspic_args:nnn { #1 } { #2 } { #3 }
4064        }
4065    }
```

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".

```
4066  \cs_new_protected:Npn \__enumext_anspic_body_dim:n #1
4067    {
4068      \bool_if:NF \l__enumext_anspic_label_above_bool
4069        {
4070          \IfDocumentMetadataTF
4071            {
4072              \tag_suspend:n {keyanspic}
4073            } { }
4074          \vbox_set:Nn \l__enumext_anspic_body_box { #1 }
4075          \dim_set:Nn \l__enumext_anspic_body_htdp_dim
4076            {
4077              \box_ht_plus_dp:N \l__enumext_anspic_body_box
4078            }
4079          \IfDocumentMetadataTF
4080            {
4081              \tag_resume:n {keyanspic}
4082            } { }
4083        }
4084    }
```

The `\__enumext_anspic_label:nn` function will process inside `\makebox` the *starred argument* '`*`' and *optional argument* passed to the command. Here we will store the ⟨*label*⟩ and *optional argument* in *prop list* and *sequence* and execute the `show-ans`, `show-pos`, `font`, `wrap-label` and `wrap-opt` keys.

```
4085  \cs_new_protected:Npn \__enumext_anspic_label:nn #1 #2
4086    {
4087      \makebox[ \l__enumext_anspic_mini_width_dim ][ c ]
4088        {
4089          \bool_if:nT { #1 }
4090            {
4091              \__enumext_keyans_addto_prop:n { #2 }
4092              \__enumext_keyans_store_ref:
4093              \__enumext_keyans_addto_seq:n { #2 }
4094              \int_gincr:N \g__enumext_check_starred_cmd_int
4095              \bool_lazy_or:nnT
4096                { \bool_if_p:N \l__enumext_show_answer_bool }
4097                { \bool_if_p:N \l__enumext_show_position_bool }
4098                {
4099                  \tl_set_eq:NN \l__enumext_label_v_tl \l__enumext_label_vi_tl
4100                  \__enumext_keyans_show_left:n { #2 }
4101                  \tl_set_eq:NN \l__enumext_label_vi_tl \l__enumext_label_v_tl
4102                }
4103            }
4104          \tl_use:N \l__enumext_label_font_style_v_tl
4105          \__enumext_wrapper_label_v:n { \l__enumext_label_vi_tl }
4106          \__enumext_keyans_show_item_opt:
```

```
4107            }
4108       }
```

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.

```
4109   \cs_new_protected:Npn \__enumext_anspic_label_pos:nnn #1 #2 #3
4110     {
4111       \stepcounter { enumXvi }
4112       \__enumext_anspic_body_dim:n { #3 }
4113       \bool_if:NTF \l__enumext_anspic_label_above_bool
4114         {
4115           \__enumext_anspic_label:nn { #1 } { #2 }
4116         }
4117         {
4118           \raisebox
4119             {
4120               -\dim_eval:n
4121                 {
4122                   \l__enumext_anspic_label_htdp_dim
4123                   + \l__enumext_anspic_body_htdp_dim
4124                   + \box_dp:N \strutbox
4125                   + \l__enumext_anspic_label_sep_skip
4126                 }
4127             }
4128             [ 0pt ] [ 0pt ]
4129             {
4130               \__enumext_anspic_label:nn { #1 } { #2 }
4131             }
4132         }
4133     }
4134   %
```

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.

```
4135   \cs_new_protected:Nn \__enumext_anspic_args:nnn
4136     {
4137       \__enumext_anspic_start_list_tag:
4138       \__enumext_anspic_label_pos:nnn { #1 } { #2 } { #3 }
4139       \__enumext_anspic_stop_start_list_tag:
4140       \bool_if:NTF \l__enumext_anspic_label_above_bool
4141         {
4142           \\[\l__enumext_anspic_label_sep_skip] #3
4143         }
4144         {
4145           \\ #3
4146         }
4147       \__enumext_anspic_stop_list_tag:
4148     }
```

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.

```
4149   \cs_new_protected:Nn \__enumext_anspic_print:n
4150     {
4151       \clist_map_function:nN { #1 } \__enumext_anspic_row:n
4152     }
4153   \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.

```
4154   \cs_new_protected:Nn \__enumext_anspic_row:n
4155     {
4156       \dim_set:Nn \l__enumext_anspic_mini_width_dim { \linewidth / #1 }
4157       \int_set:Nn \l__enumext_anspic_above_int { \l__enumext_anspic_below_int }
4158       \int_set:Nn \l__enumext_anspic_below_int { \l__enumext_anspic_above_int + #1 }
4159       \int_step_inline:nnn
4160         { \l__enumext_anspic_above_int + 1 }
4161         { \l__enumext_anspic_below_int }
4162         {
4163           \IfDocumentMetadataTF
4164             {
4165               \tag_suspend:n {minipage}
```

```
4166          } { }
4167        \begin{minipage}[ \l__enumext_anspic_mini_pos_str ]{ \l__enumext_anspic_mini_width_dim }
4168          \centering
4169          \seq_item:Nn \l__enumext_anspic_args_seq { ##1 }
4170        \end{minipage}
4171        \IfDocumentMetadataTF
4172          {
4173            \tag_resume:n {minipage}
4174          } { }
4175      }
4176    \par
4177  }
```

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*.

```
4178  \cs_new_protected:Nn \__enumext_anspic_exec:
4179    {
4180      \tl_if_empty:NTF \l__enumext_anspic_layout_style_tl
4181        {
4182          \__enumext_anspic_print:e { \seq_count:N \l__enumext_anspic_args_seq }
4183        }
4184        {
4185          \__enumext_anspic_print:V \l__enumext_anspic_layout_style_tl
4186        }
4187    }
```

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

13.42   The horizontal environments

Generating *horizontal list environments* is NOT as simple as standard LaTeX list environments. The fundamental part of the code is adapted from the shortlst package to a more modern version using expl3. It is not possible to redefine \item and \makelabel using \RenewDocumentCommand as in the vertical *non starred* versions.

To achieve the *horizontal list environments* we will capture the \item command and the ⟨*content*⟩ of this in *horizontal box* using \makebox for the label and a minipage environment for the ⟨*content*⟩ passed to \item, we will also add the *optional argument* (⟨*number*⟩) to \item to be able to *join columns* horizontally, in simple terms, we want \item to behave in the same way as in the enumext environment but adding an *first optional argument* (⟨*number*⟩).

A side effect is the limitation of using \item in this way *without* using \RenewDocumentCommand, which loses the original definition and affects the *standard list environments* provided by LaTeX and any environment defined using base list environment, including: itemize, enumerate, description, quote, quotation, verse, center, flushleft, flushright, verbatim, tabbing, trivlist, list and all environments created with \newtheorem.

💣 One way to get around this is to use something like:

    \AddToHook{env/enumerate/before}{recover original \item definition}

inside minipage, but in my partial tests this does not have the desired effect and the vertical and horizontal spacing is distorted. For now this will remain as a limitation and I will see if it is feasible to implement it in the future.

🏷 For compatibility with the *tagged* PDF we close the environments according to the presence or not of the mini-env key.

13.42.1   Functions for item box width

__enumext_starred_columns_set_vii:
__enumext_starred_columns_set_viii:

We set the default value for the *width of the box* containing the ⟨*content*⟩ of the items for enumext* environment.

```
4188  \cs_new_protected:Nn \__enumext_starred_columns_set_vii:
4189    {
4190      \dim_compare:nNnT { \l__enumext_columns_sep_vii_dim } = { \c_zero_dim }
4191        {
4192          \dim_set:Nn \l__enumext_columns_sep_vii_dim
4193            {
4194              ( \l__enumext_labelwidth_vii_dim + \l__enumext_labelsep_vii_dim )
4195              / \l__enumext_columns_vii_int
4196            }
4197        }
4198      \int_set:Nn \l__enumext_tmpa_vii_int { \l__enumext_columns_vii_int - 1 }
4199      \dim_set:Nn \l__enumext_item_width_vii_dim
4200        {
4201          ( \linewidth - \l__enumext_columns_sep_vii_dim * \l__enumext_tmpa_vii_int )
4202          / \l__enumext_columns_vii_int
4203          - \l__enumext_labelwidth_vii_dim
```

```
4204          - \l__enumext_labelsep_vii_dim
4205        }
```

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

```
4206      \dim_compare:nNnT { \l__enumext_rightmargin_vii_dim } > { \c_zero_dim }
4207        {
4208          \dim_sub:Nn \l__enumext_item_width_vii_dim
4209            {
4210              ( \l__enumext_rightmargin_vii_dim * \l__enumext_tmpa_vii_int )
4211              / \l__enumext_columns_vii_int
4212            }
4213          \dim_add:Nn \l__enumext_columns_sep_vii_dim
4214            {
4215              \l__enumext_rightmargin_vii_dim
4216            }
4217        }
4218    }
```

Same implementation for the `keyans*` environment.

```
4219  \cs_new_protected:Nn \__enumext_starred_columns_set_viii:
4220    {
4221      \dim_compare:nNnT { \l__enumext_columns_sep_viii_dim } = { \c_zero_dim }
4222        {
4223          \dim_set:Nn \l__enumext_columns_sep_viii_dim
4224            {
4225              ( \l__enumext_labelwidth_viii_dim + \l__enumext_labelsep_viii_dim )
4226              / \l__enumext_columns_viii_int
4227            }
4228        }
4229      \int_set:Nn \l__enumext_tmpa_viii_int { \l__enumext_columns_viii_int - 1 }
4230      \dim_set:Nn \l__enumext_item_width_viii_dim
4231        {
4232          ( \linewidth - \l__enumext_columns_sep_viii_dim * \l__enumext_tmpa_viii_int )
4233          / \l__enumext_columns_viii_int
4234          - \l__enumext_labelwidth_viii_dim
4235          - \l__enumext_labelsep_viii_dim
4236        }
4237      \dim_compare:nNnT { \l__enumext_rightmargin_viii_dim } > { \c_zero_dim }
4238        {
4239          \dim_sub:Nn \l__enumext_item_width_viii_dim
4240            {
4241              ( \l__enumext_rightmargin_viii_dim * \l__enumext_tmpa_vii_int )
4242              / \l__enumext_columns_viii_int
4243            }
4244          \dim_add:Nn \l__enumext_columns_sep_viii_dim
4245            {
4246              \l__enumext_rightmargin_viii_dim
4247            }
4248        }
4249    }
```

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

13.42.2 Functions for join item columns

`\__enumext_starred_joined_item_vii:n`
`\__enumext_starred_joined_item_viii:n`

The functions `\__enumext_starred_joined_item_vii:n` and `\__enumext_starred_joined_item_-viii:n` will set the *width* of the box in which the ⟨*content*⟩ passed to `\item`(⟨*columns*⟩) will be stored together with the value of `\itemwidth` for the `enumext*` environment.

```
4250  \cs_new_protected:Npn \__enumext_starred_joined_item_vii:n #1
4251    {
4252      \int_set:Nn \l__enumext_joined_item_vii_int {#1}
4253      \int_compare:nNnT { \l__enumext_joined_item_vii_int } > { \l__enumext_columns_vii_int }
4254        {
4255          \msg_warning:nnee { enumext } { item-joined }
4256            { \int_use:N \l__enumext_joined_item_vii_int }
4257            { \int_use:N \l__enumext_columns_vii_int }
4258          \int_set:Nn \l__enumext_joined_item_vii_int
4259            {
4260              \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + 1
4261            }
4262        }
4263      \int_compare:nNnT
4264        { \l__enumext_joined_item_vii_int }
```

```
4265          >
4266        { \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + 1 }
4267        {
4268          \msg_warning:nnee { enumext } { item-joined-columns }
4269            { \int_use:N \l__enumext_joined_item_vii_int }
4270            {
4271              \int_eval:n
4272                { \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + 1 }
4273            }
4274          \int_set:Nn \l__enumext_joined_item_vii_int
4275            {
4276              \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + 1
4277            }
4278        }
4279      \int_compare:nNnTF { \l__enumext_joined_item_vii_int } > { 1 }
4280        {
4281          \int_set_eq:NN \l__enumext_joined_item_aux_vii_int \l__enumext_joined_item_vii_int
4282          \int_decr:N \l__enumext_joined_item_aux_vii_int
4283          \int_add:Nn \l__enumext_item_column_pos_vii_int { \l__enumext_joined_item_aux_vii_int }
4284          \int_gadd:Nn \g__enumext_item_count_all_vii_int { \l__enumext_joined_item_aux_vii_int }
4285          \dim_set:Nn \l__enumext_joined_width_vii_dim
4286            {
4287              \l__enumext_item_width_vii_dim * \l__enumext_joined_item_vii_int
4288              + (  \l__enumext_labelwidth_vii_dim + \l__enumext_labelsep_vii_dim
4289                 + \l__enumext_columns_sep_vii_dim
4290              )*\l__enumext_joined_item_aux_vii_int
4291            }
4292          \dim_set_eq:NN \itemwidth \l__enumext_joined_width_vii_dim
4293        }
4294        {
4295          \dim_set_eq:NN \l__enumext_joined_width_vii_dim \l__enumext_item_width_vii_dim
4296          \dim_set_eq:NN \itemwidth \l__enumext_item_width_vii_dim
4297        }
4298    }
```

Same implementation for the keyans* environment.

```
4299  \cs_new_protected:Npn \__enumext_starred_joined_item_viii:n #1
4300    {
4301      \int_set:Nn \l__enumext_joined_item_viii_int {#1}
4302      \int_compare:nNnT { \l__enumext_joined_item_viii_int } > { \l__enumext_columns_viii_int }
4303        {
4304          \msg_warning:nnee { enumext } { item-joined }
4305            { \int_use:N \l__enumext_joined_item_viii_int }
4306            { \int_use:N \l__enumext_columns_viii_int }
4307          \int_set:Nn \l__enumext_joined_item_viii_int
4308            {
4309              \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1
4310            }
4311        }
4312      \int_compare:nNnT
4313        { \l__enumext_joined_item_viii_int }
4314          >
4315        { \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1 }
4316        {
4317          \msg_warning:nnee { enumext } { item-joined-columns }
4318            { \int_use:N \l__enumext_joined_item_viii_int }
4319            {
4320              \int_eval:n
4321                { \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1 }
4322            }
4323          \int_set:Nn \l__enumext_joined_item_viii_int
4324            {
4325              \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1
4326            }
4327        }
4328      \int_compare:nNnTF { \l__enumext_joined_item_viii_int } > { 1 }
4329        {
4330          \int_set_eq:NN \l__enumext_joined_item_aux_viii_int \l__enumext_joined_item_viii_int
4331          \int_decr:N \l__enumext_joined_item_aux_viii_int
4332          \int_add:Nn \l__enumext_item_column_pos_viii_int { \l__enumext_joined_item_aux_viii_int }
4333          \int_gadd:Nn \g__enumext_item_count_all_viii_int { \l__enumext_joined_item_aux_viii_int }
4334          \dim_set:Nn \l__enumext_joined_width_viii_dim
```

```
4335              {
4336                \l__enumext_item_width_viii_dim * \l__enumext_joined_item_viii_int
4337                  + ( \l__enumext_labelwidth_viii_dim + \l__enumext_labelsep_viii_dim
4338                    + \l__enumext_columns_sep_viii_dim
4339                  )*\l__enumext_joined_item_aux_viii_int
4340              }
4341            \dim_set_eq:NN \itemwidth \l__enumext_joined_width_viii_dim
4342          }
4343          {
4344            \dim_set_eq:NN \l__enumext_joined_width_viii_dim \l__enumext_item_width_viii_dim
4345            \dim_set_eq:NN \itemwidth \l__enumext_item_width_viii_dim
4346          }
4347      }
```

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

13.42.3   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.

```
4348 \cs_new_protected:Nn \__enumext_start_mini_vii:
4349    {
4350      \dim_compare:nNnT { \l__enumext_minipage_right_vii_dim } > { \c_zero_dim }
4351        {
4352          \dim_set:Nn \l__enumext_minipage_left_vii_dim
4353            {
4354              \linewidth
4355              - \l__enumext_minipage_right_vii_dim
4356              - \l__enumext_minipage_hsep_vii_dim
4357            }
4358          \bool_set_true:N \l__enumext_minipage_active_vii_bool
4359          \dim_gset_eq:NN
4360            \g__enumext_minipage_right_vii_dim
4361            \l__enumext_minipage_right_vii_dim
4362          \__enumext_mini_addvspace_vii:
4363          \nointerlineskip\noindent
4364          \__enumext_mini_page{ \l__enumext_minipage_left_vii_dim }
4365        }
4366    }
```

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 (§13.43).

```
4367 \cs_new_protected:Nn \__enumext_stop_mini_vii:
4368    {
4369      \bool_if:NTF \l__enumext_minipage_active_vii_bool
4370        {
4371          \__enumext_stop_list:
4372          \__enumext_stop_store_level_vii:
4373          \IfDocumentMetadataTF { \tag_resume:n {enumext*} } { }
4374          \end__enumext_mini_page
4375          \hfill
4376          \bool_gset_true:N \g__enumext_minipage_active_vii_bool
4377        }
4378        {
4379          \__enumext_stop_list:
4380          \__enumext_stop_store_level_vii:
4381        }
4382    }
```

(*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.

```
4383  \__enumext_after_env:nn {enumext*}
4384    {
4385      \bool_if:NT \g__enumext_minipage_active_vii_bool
4386        {
4387          \__enumext_minipage:w [ t ] { \g__enumext_minipage_right_vii_dim }
4388            \legacy_if_gset_false:n { @minipage }
4389            \skip_vertical:N \c_zero_skip
4390            \par\addvspace { \g__enumext_minipage_right_skip }
4391            \bool_if:NF \g__enumext_minipage_center_vii_bool
4392              {
4393                \tl_put_left:Nn \g__enumext_miniright_code_vii_tl
4394                  {
4395                    \centering
4396                  }
4397              }
4398            \vbox_set_top:Nn \l__enumext_miniright_code_vii_box
4399              {
4400                \tl_use:N \g__enumext_miniright_code_vii_tl
4401              }
4402            \box_use_drop:N \l__enumext_miniright_code_vii_box
4403            \skip_vertical:N \c_zero_skip
4404          \__enumext_endminipage:
4405          \par\addvspace{ \g__enumext_minipage_after_skip }
4406        }
4407      \bool_gset_false:N \g__enumext_minipage_active_vii_bool
4408      \bool_gset_true:N \g__enumext_minipage_center_vii_bool
4409      \tl_gclear:N \g__enumext_miniright_code_vii_tl
4410      \dim_gzero:N \g__enumext_minipage_right_vii_dim
4411      \bool_gset_false:N \g__enumext_starred_bool
4412    }
```

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

```
4413  \cs_new_protected:Nn \__enumext_start_mini_viii:
4414    {
4415      \dim_compare:nNnT { \l__enumext_minipage_right_viii_dim } > { \c_zero_dim }
4416        {
4417          \dim_set:Nn \l__enumext_minipage_left_viii_dim
4418            {
4419              \linewidth
4420              - \l__enumext_minipage_right_viii_dim
4421              - \l__enumext_minipage_hsep_viii_dim
4422            }
4423          \bool_set_true:N \l__enumext_minipage_active_viii_bool
4424          \dim_gset_eq:NN
4425            \g__enumext_minipage_right_viii_dim
4426            \l__enumext_minipage_right_viii_dim
4427          \__enumext_mini_addvspace_viii:
4428          \nointerlineskip\noindent
4429          \__enumext_mini_page{ \l__enumext_minipage_left_viii_dim }
4430        }
4431    }
4432  \cs_new_protected:Nn \__enumext_stop_mini_viii:
4433    {
4434      \bool_if:NTF \l__enumext_minipage_active_viii_bool
4435        {
4436          \__enumext_stop_list:
4437          \IfDocumentMetadataTF { \tag_resume:n {keyans*} } { }
4438          \end__enumext_mini_page
4439          \hfill
4440          \bool_gset_true:N \g__enumext_minipage_active_viii_bool
4441        }
4442        {
4443          \__enumext_stop_list:
4444        }
4445    }
4446  \__enumext_after_env:nn {keyans*}
4447    {
4448      \bool_if:NT \g__enumext_minipage_active_viii_bool
4449        {
4450          \__enumext_mini_page{ \g__enumext_minipage_right_viii_dim }
```

```
4451            \par\addvspace { \g__enumext_minipage_right_skip }
4452            \bool_if:NF \g__enumext_minipage_center_viii_bool
4453              {
4454                 \tl_put_left:Nn \g__enumext_miniright_code_viii_tl
4455                   {
4456                      \centering
4457                   }
4458              }
4459            \vbox_set_top:Nn \l__enumext_miniright_code_viii_box
4460              {
4461                 \tl_use:N \g__enumext_miniright_code_viii_tl
4462              }
4463            \box_use_drop:N \l__enumext_miniright_code_viii_box
4464          \end__enumext_mini_page
4465          \par\addvspace{ \g__enumext_minipage_after_skip }
4466        }
4467      \bool_gset_false:N \g__enumext_minipage_active_viii_bool
4468      \bool_gset_true:N \g__enumext_minipage_center_viii_bool
4469      \tl_gclear:N \g__enumext_miniright_code_viii_tl
4470      \dim_gzero:N \g__enumext_minipage_right_viii_dim
4471    }
```

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

13.42.4 Redefining \footnote command

__enumext_footnotetext:nn
__enumext_renew_footnote:
__enumext_print_footnote:

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

```
4472 \cs_new_protected:Nn \__enumext_footnotetext:nn
4473    {
4474      \footnotetext[#1]{#2}
4475    }
4476 \cs_new_protected:Nn \__enumext_renew_footnote:
4477    {
4478      \seq_gclear:N \g__enumext_footnote_arg_seq
4479      \seq_gclear:N \g__enumext_footnote_int_seq
4480      \RenewDocumentCommand \footnote { o +m }
4481        {
4482          \tl_if_novalue:nTF {##1}
4483            {
4484              \stepcounter{footnote}
4485              \int_gset_eq:Nc \g__enumext_footnote_int { c@footnote }
4486            }
4487            {
4488              \int_gset:Nn \g__enumext_footnote_int { ##1 }
4489            }
4490          \footnotemark [ \g__enumext_footnote_int ]
4491          \seq_gput_right:Nn \g__enumext_footnote_arg_seq { ##2 }
4492          \seq_gput_right:NV \g__enumext_footnote_int_seq \g__enumext_footnote_int
4493        }
4494    }
4495 \cs_new_protected:Nn \__enumext_print_footnote:
4496    {
4497      \seq_if_empty:NF \g__enumext_footnote_int_seq
4498        {
4499          \seq_map_pairwise_function:NNN
4500            \g__enumext_footnote_int_seq
4501            \g__enumext_footnote_arg_seq
4502            \__enumext_footnotetext:nn
4503        }
4504    }
```

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

13.43 The environment enumext*

enumext* First we will generate the environment and we will give a temporary definition to __enumext_stop_-item_tmp_vii: equal to __enumext_first_item_tmp_vii: and next to \item equal to __enumext_-start_item_tmp_vii: which we will redefine later. Unlike the implementation used by the shortlst package, we will not set the values of \rightskip and \@rightskip equal to \@flushglue whose value is

`0.0pt plus 1.0 fil`, in the tests I have performed this fails in some circumstances and different results are obtained when using pdfTeX and LuaTeX.

```
4505  \NewDocumentEnvironment{enumext*}{ o }
4506    {
4507      \__enumext_safe_exec_vii:
4508      \__enumext_parse_keys_vii:n {#1}
4509      \__enumext_before_list_vii:
4510      \__enumext_start_store_level_vii:
4511      \__enumext_start_list:nn { }
4512        {
4513          \__enumext_list_arg_two_vii:
4514          \__enumext_before_keys_exec_vii:
4515        }
4516      \IfDocumentMetadataTF { \tag_suspend:n {enumext*} } { }
4517      \__enumext_starred_columns_set_vii:
4518      \item[] \scan_stop:
4519      \cs_set_eq:NN \__enumext_stop_item_tmp_vii: \__enumext_first_item_tmp_vii:
4520      \cs_set_eq:NN \item \__enumext_start_item_tmp_vii:
4521      \ignorespaces
4522    }
4523    {
4524      \IfDocumentMetadataTF { \tag_struct_end:n {tag=text-unit} } { }
4525      \__enumext_stop_item_tmp_vii:
4526      \__enumext_remove_extra_parsep_vii:
4527      \__enumext_after_list_vii:
4528    }
```

(*End of definition for* enumext*. *This function is documented on page 5.*)

`\__enumext_safe_exec_vii:` We will first call the function `\__enumext_internal_mini_page:` to create the environment `__enumext_-mini_page`, then the function `\__enumext_is_not_nested:` which sets `\g__enumext_starred_bool` to true if we are not nested within enumext, we will increment `\l__enumext_level_h_int` to restrict nesting of the environment, set `\l__enumext_starred_bool` to true and finally call the function `\__enumext_is_-on_first_level:` which sets `\l__enumext_starred_first_bool` to true if we are not nested, allowing the *"storage system"* to be used.

```
4529  \cs_new_protected:Nn \__enumext_safe_exec_vii:
4530    {
4531      \__enumext_internal_mini_page:
4532      \__enumext_is_not_nested:
4533      \int_incr:N \l__enumext_level_h_int
4534      \int_compare:nNnT { \l__enumext_level_h_int } > { 1 }
4535        {
4536          \msg_error:nn { enumext } { nested }
4537        }
4538      \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
4539        {
4540          \msg_error:nnn { enumext } { nested-horizontal } { keyans*}
4541        }
4542      \bool_set_true:N \l__enumext_starred_bool
4543      \bool_set_false:N \l__enumext_standar_bool
4544      \__enumext_is_on_first_level:
4545    }
```

(*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.

```
4546  \cs_new_protected:Npn \__enumext_parse_keys_vii:n #1
4547    {
4548      \tl_if_novalue:nF {#1}
4549        {
4550          \str_clear:N \l__enumext_series_str
4551          \keys_set:nn { enumext / enumext* } {#1}
4552          \__enumext_parse_series:n {#1}
4553          \__enumext_store_active_keys_vii:n {#1}
4554        }
4555    }
```

(*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*.

```
4556 \cs_new_protected:Nn \__enumext_before_list_vii:
4557   {
4558     \__enumext_vspace_above_vii:
4559     \__enumext_check_ans_active:
4560     \__enumext_before_args_exec_vii:
4561     \__enumext_start_mini_vii:
4562   }
```

(*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:` (§13.42.3) used by the keys mini-env, mini-right and mini-right*, then to the functions `\__enumext_after_stop_-list_vii:` used by the key after, `\__enumext_check_ans_key_hook:` used by the key check-ans, `\__enumext_vspace_below_vii:` used by the keys below and below*. Finally set `\l__enumext_-starred_bool` to false and call the `\__enumext_resume_save_counter:` function used by the series, resume and resume* keys.

```
4563 \cs_new_protected:Nn \__enumext_after_list_vii:
4564   {
4565     \__enumext_stop_mini_vii:
4566     \__enumext_after_stop_list_vii:
4567     \__enumext_check_ans_key_hook:
4568     \__enumext_vspace_below_vii:
4569     \bool_set_false:N \l__enumext_starred_bool
4570     \__enumext_resume_save_counter:
4571   }
```

(*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.

```
4572 \cs_new_protected:Nn \__enumext_start_store_level_vii:
4573   {
4574     \bool_if:NT \l__enumext_store_active_bool
4575       {
4576         \int_compare:nNnT { \l__enumext_level_int } > { 0 }
4577           {
4578             \__enumext_store_level_open_vii:
4579           }
4580       }
4581   }
4582 \cs_new_protected:Nn \__enumext_stop_store_level_vii:
4583   {
4584     \bool_if:NT \l__enumext_store_active_bool
4585       {
4586         \int_compare:nNnT { \l__enumext_level_int } > { 0 }
4587           {
4588             \__enumext_store_level_close_vii:
4589           }
4590       }
4591   }
```

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

13.43.1 The command \item in enumext*

`\__enumext_first_item_tmp_vii:` The `\__enumext_first_item_tmp_vii:` function will remove horizontal space equal to `\labelwidth` plus `\labelsep` to the left of the *"first"* `\item` in the environment at the point of execution of this function, where it is equal to the `\__enumext_stop_item_tmp_vii:` function inside the environment body definition.

```
4592 \cs_new_protected_nopar:Nn \__enumext_first_item_tmp_vii:
4593   {
4594     \skip_horizontal:n
4595       {
4596         -\l__enumext_labelwidth_vii_dim - \l__enumext_labelsep_vii_dim
```

```
4597          }
4598       \ignorespaces
4599    }
```

(*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.

```
4600  \cs_new_protected_nopar:Nn \__enumext_start_item_tmp_vii:
4601    {
4602       \__enumext_stop_item_tmp_vii:
4603       \int_incr:N \l__enumext_item_column_pos_vii_int
4604       \int_gincr:N \g__enumext_item_count_all_vii_int
4605       \__enumext_item_peek_args_vii:
4606    }
```

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).

```
4607  \cs_new_protected:Nn \__enumext_item_peek_args_vii:
4608    {
4609       \peek_meaning:NTF (
4610          { \__enumext_joined_item_vii:w }
4611          { \__enumext_joined_item_vii:w (1) }
4612    }
```

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.

```
4613  \cs_new_protected:Npn \__enumext_joined_item_vii:w (#1)
4614    {
4615       \__enumext_starred_joined_item_vii:n {#1}
4616       \peek_meaning_remove:NTF *
4617          { \__enumext_starred_item_vii:w  }
4618          { \__enumext_standar_item_vii:w }
4619    }
```

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].

```
4620  \cs_new_protected:Npn \__enumext_standar_item_vii:w
4621    {
4622       \bool_set_false:N \l__enumext_item_starred_vii_bool
4623       \peek_meaning:NTF [
4624          {
4625             \bool_set_eq:NN \l__enumext_wrap_label_vii_bool \l__enumext_wrap_label_opt_vii_bool
4626             \__enumext_start_item_vii:w
4627          }
4628          {
4629             \bool_set_true:N \l__enumext_wrap_label_vii_bool
4630             \legacy_if_set_true:n { @noitemarg }
4631             \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ] \ignorespaces
4632          }
4633    }
```

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⟩].

```
4634  \cs_new_protected:Npn \__enumext_starred_item_vii:w
4635    {
4636       \bool_set_true:N \l__enumext_item_starred_vii_bool
4637       \bool_set_true:N \l__enumext_wrap_label_vii_bool
4638       \peek_meaning:NTF [
4639          { \__enumext_starred_item_vii_aux_i:w }
```

```
4640            { \__enumext_starred_item_vii_aux_ii:w }
4641      }
4642  \cs_new_protected:Npn \__enumext_starred_item_vii_aux_i:w [#1]
4643      {
4644        \tl_gset:Nn \g__enumext_item_symbol_aux_vii_tl {#1}
4645        \__enumext_starred_item_vii_aux_ii:w
4646      }
4647  \cs_new_protected:Npn \__enumext_starred_item_vii_aux_ii:w
4648      {
4649        \peek_meaning:NTF [
4650            { \__enumext_starred_item_vii_aux_iii:w }
4651            {
4652              \dim_set_eq:NN \l__enumext_item_symbol_sep_vii_dim \l__enumext_labelsep_vii_dim
4653              \legacy_if_set_true:n { @noitemarg }
4654              \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ] \ignorespaces
4655            }
4656      }
4657  \cs_new_protected:Npn \__enumext_starred_item_vii_aux_iii:w [#1]
4658      {
4659        \dim_set:Nn \l__enumext_item_symbol_sep_vii_dim {#1}
4660        \legacy_if_set_true:n { @noitemarg }
4661        \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ] \ignorespaces
4662      }
```

(*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

```
4663  \cs_new_protected_nopar:Npn \__enumext_fake_make_label_vii:n #1
4664      {
4665        \legacy_if:nT { @noitemarg }
4666            {
4667              \legacy_if_set_false:n { @noitemarg }
4668              \legacy_if:nT { @nmbrlist }
4669                {
4670                  \IfDocumentMetadataTF
4671                    {
4672                      \bool_if:NT \l__enumext_hyperref_bool
4673                        {
4674                          \legacy_if_set_true:n { @hyper@item }
4675                        }
4676                    } { }
4677                  \refstepcounter{enumXvii}
4678                  \bool_if:NT \l__enumext_check_answers_bool
4679                    {
4680                      \int_gincr:N \g__enumext_item_number_int
4681                      \bool_set_true:N \l__enumext_item_number_bool
4682                    }
4683                }
4684            }
4685        \bool_if:NT \l__enumext_item_starred_vii_bool
4686            {
4687              \tl_if_blank:VT \g__enumext_item_symbol_aux_vii_tl
4688                {
4689                  \tl_gset_eq:NN
4690                    \g__enumext_item_symbol_aux_vii_tl \l__enumext_item_symbol_vii_tl
4691                }
4692              \mode_leave_vertical:
4693              \skip_horizontal:n { -\l__enumext_item_symbol_sep_vii_dim }
4694              \hbox_overlap_left:n { \g__enumext_item_symbol_aux_vii_tl }
4695              \skip_horizontal:N \l__enumext_item_symbol_sep_vii_dim
```

```
4696        \tl_gclear:N \g__enumext_item_symbol_aux_vii_tl
4697      }
4698    \makebox[ \l__enumext_labelwidth_vii_dim ][ \l__enumext_align_label_vii_str ]
4699      {
4700        \tl_use:N \l__enumext_label_font_style_vii_tl
4701        \bool_if:NTF \l__enumext_wrap_label_vii_bool
4702          {
4703            \__enumext_wrapper_label_vii:n {#1}
4704          }
4705          { #1 }
4706      }
4707    \skip_horizontal:N \l__enumext_labelsep_vii_dim \ignorespaces
4708    }
```

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

13.43.2 Real definition of \item in enumext*

The functions `\__enumext_start_item_vii:w` and `\__enumext_stop_item_vii:` executing the true definition of `\item` inside the enumext* environment, unlike the implementation in shortlst we will NOT use an extra group and the plain form of the lrbox environment.

`\__enumext_start_item_vii:w`
`\__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`.

```
4709  \cs_new_protected_nopar:Npn \__enumext_start_item_vii:w [#1]
4710    {
4711      \cs_set_eq:NN \__enumext_stop_item_tmp_vii: \__enumext_stop_item_vii:
4712      \hbox_set_to_wd:Nnw \l__enumext_item_text_vii_box
4713        {
4714          \l__enumext_joined_width_vii_dim
4715          + \l__enumext_labelwidth_vii_dim
4716          + \l__enumext_labelsep_vii_dim
4717        }
```

If `\DocumentMetadata` is not active and the state of the variable `\l__enumext_footnotes_key_bool` is false, we will redefine the `\footnote` command.

```
4718      \IfDocumentMetadataTF { }
4719        {
4720          \bool_if:NF \l__enumext_footnotes_key_bool
4721            {
4722              \__enumext_renew_footnote:
4723            }
4724        }
```

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

```
4725      \__enumext_start_list_tag:n {enumext*}
4726      \__enumext_fake_make_label_vii:n {#1}
4727      \__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.

```
4728      \__enumext_minipage:w [ t ]{ \l__enumext_joined_width_vii_dim  }
4729        \dim_set_eq:NN \parindent \l__enumext_listparindent_vii_dim
4730        \skip_set_eq:NN \parskip \l__enumext_parsep_vii_skip
4731        \__enumext_unskip_unkern:
4732        \__enumext_unskip_unkern:
4733        \skip_horizontal:n { -\l__enumext_listparindent_vii_dim } \ignorespaces
4734        \tl_use:N \l__enumext_fake_item_indent_vii_tl
4735        \tl_use:N \l__enumext_after_list_args_vii_tl
4736    }
```

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*.

```
4737  \cs_new_protected_nopar:Nn \__enumext_stop_item_vii:
4738    {
4739      \__enumext_endminipage:
4740    \__enumext_stop_list_tag:n {enumext*}
4741    \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* and *footnotes* if it was redefined.

```
4742      \int_set:Nn \hbadness { 10000 }
4743      \box_use_drop:N \l__enumext_item_text_vii_box
4744      \IfDocumentMetadataTF { }
4745        {
4746          \bool_if:NF \l__enumext_footnotes_key_bool
4747            {
4748              \__enumext_print_footnote:
4749            }
4750        }
```

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`.

```
4751      \int_compare:nNnTF
4752        { \l__enumext_item_column_pos_vii_int } = { \l__enumext_columns_vii_int }
4753        {
4754          \par\noindent
4755          \int_zero:N \l__enumext_item_column_pos_vii_int
4756        }
4757        {
4758          \skip_horizontal:N \l__enumext_columns_sep_vii_dim
4759        }
4760      }
```

(*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*⟩.

```
4761 \cs_new_protected:Nn \__enumext_remove_extra_parsep_vii:
4762   {
4763      \int_compare:nNnT
4764        {
4765          \int_mod:nn
4766            {  \g__enumext_item_count_all_vii_int } { \l__enumext_columns_vii_int }
4767        }
4768        =
4769        { 0 }
4770        {
4771          \para_end:
4772          \skip_vertical:n { -\l__enumext_itemsep_vii_skip }
4773          \skip_vertical:N \c_zero_skip
4774          \int_gzero:N \g__enumext_item_count_all_vii_int
4775        }
4776   }
```

(*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`.

```
4777 \__enumext_after_env:nn {enumext*}
4778   {
4779      \__enumext_execute_after_env:
4780   }
```

13.44   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.

```
4781 \NewDocumentEnvironment{keyans*}{ o }
4782   {
4783      \__enumext_safe_exec_viii:
4784      \__enumext_parse_keys_viii:n {#1}
4785      \__enumext_before_list_viii:
4786      \__enumext_start_list:nn { }
4787        {
4788          \__enumext_list_arg_two_viii:
4789          \__enumext_before_keys_exec_viii:
4790        }
```

```
4791    \IfDocumentMetadataTF { \tag_suspend:n {keyans*} } { }
4792    \__enumext_starred_columns_set_viii:
4793    \item[] \scan_stop:
4794    \cs_set_eq:NN \__enumext_stop_item_tmp_viii: \__enumext_first_item_tmp_viii:
4795    \cs_set_eq:NN \item \__enumext_start_item_tmp_viii:
4796    \ignorespaces
4797  }
4798  {
4799    \IfDocumentMetadataTF { \tag_struct_end:n {tag=text-unit} } { }
4800    \__enumext_stop_item_tmp_viii:
4801    \__enumext_remove_extra_parsep_viii:
4802    \__enumext_check_starred_cmd:n { item }
4803    \__enumext_after_list_viii:
4804  }
```

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

__enumext_safe_exec_viii: The __enumext_safe_exec_viii: function will first check if the save-ans key is active and only when this is true the environment will be available, it will increment the value of \l__enumext_keyans_level_h_int and return an error message when we are nesting the environment, then it will call the __enumext_-keyans_name_and_start: function in charge of saving the name of the environment and the line it is running on, then it will check if we are trying to nest keyans* in enumext* returning an error and we will set \l__enumext_starred_bool to true, finally we will check if we are within the appropriate level within the enumext environment.

```
4805  \cs_new_protected:Nn \__enumext_safe_exec_viii:
4806    {
4807      \bool_if:NF \l__enumext_store_active_bool
4808        {
4809          \msg_error:nnnn { enumext } { wrong-place }{ keyans* }{ save-ans }
4810        }
4811      \int_incr:N \l__enumext_keyans_level_h_int
4812      \int_compare:nNnT { \l__enumext_keyans_level_h_int } > { 1 }
4813        {
4814          \msg_error:nn { enumext } { nested }
4815        }
4816      \__enumext_keyans_name_and_start:
4817      \bool_if:NT \l__enumext_starred_bool
4818        {
4819          \msg_error:nnn { enumext } { nested-horizontal } { enumext* }
4820        }
4821      \bool_set_true:N \l__enumext_starred_bool
4822      % Set false for interfering with enumext nested in keyans* (yes, its possible and crayze)
4823      \bool_set_false:N \l__enumext_store_active_bool
4824      \int_compare:nNnT { \l__enumext_level_int } > { 1 }
4825        {
4826          \msg_error:nn { enumext } { keyans-wrong-level }
4827        }
4828    }
```

(*End of definition for* __enumext_safe_exec_viii:.)

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

```
4829  \cs_new_protected:Npn \__enumext_parse_keys_viii:n #1
4830    {
4831      \tl_if_novalue:nF {#1}
4832        {
4833          \keys_set:nn { enumext / keyans* } {#1}
4834        }
4835    }
```

(*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.

```
4836  \cs_new_protected:Nn \__enumext_before_list_viii:
4837    {
4838      \__enumext_vspace_above_viii:
4839      \__enumext_before_args_exec_viii:
4840      \__enumext_start_mini_viii:
4841    }
```

(*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.

```
4842 \cs_new_protected:Nn \__enumext_after_list_viii:
4843   {
4844     \__enumext_stop_mini_viii:
4845     \__enumext_after_stop_list_viii:
4846     \__enumext_vspace_below_viii:
4847   }
```

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

13.44.1   The command `\item` in keyans*

The idea here is to make the `\item` command behave in the same way as in the `keyans` environment with the difference of the *optional argument* (⟨*number*⟩) which works in the same way as in the `enumext*` environment. In simple terms we want to store the ⟨*label*⟩ next to the [⟨*content*⟩] if it is present in the *sequence* and *prop list* defined by `save-ans` key for `\item*`, `\item*[`⟨*content*⟩`]`, `\item(`⟨*number*⟩`)*` and `\item(`⟨*number*⟩`)*[`⟨*content*⟩`]` commands.

`\__enumext_first_item_tmp_viii:` The `\__enumext_first_item_tmp_viii:` function will remove horizontal space equal to `\labelwidth` plus `\labelsep` to the left of the *"first"* `\item` in the environment at the point of execution of this function, where it is equal to the `\__enumext_stop_item_tmp_viii:` function inside the environment body definition.

```
4848 \cs_new_protected_nopar:Nn \__enumext_first_item_tmp_viii:
4849   {
4850     \skip_horizontal:n
4851       {
4852         -\l__enumext_labelwidth_viii_dim - \l__enumext_labelsep_viii_dim
4853       }
4854     \ignorespaces
4855   }
```

(*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`.

```
4856 \cs_new_protected_nopar:Nn \__enumext_start_item_tmp_viii:
4857   {
4858     \__enumext_stop_item_tmp_viii:
4859     \int_incr:N \l__enumext_item_column_pos_viii_int
4860     \int_gincr:N \g__enumext_item_count_all_viii_int
4861     \__enumext_item_peek_args_viii:
4862   }
```

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`).

```
4863 \cs_new_protected:Nn \__enumext_item_peek_args_viii:
4864   {
4865     \peek_meaning:NTF (
4866       { \__enumext_joined_item_viii:w }
4867       { \__enumext_joined_item_viii:w (1) }
4868   }
```

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`.

```
4869 \cs_new_protected:Npn \__enumext_joined_item_viii:w (#1)
4870   {
4871     \__enumext_starred_joined_item_viii:n {#1}
4872     \peek_meaning_remove:NTF *
4873       { \__enumext_starred_item_viii:w  }
4874       { \__enumext_standar_item_viii:w }
4875   }
```

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].

```
4876 \cs_new_protected:Npn \__enumext_standar_item_viii:w
4877   {
4878     \bool_set_false:N \l__enumext_item_starred_viii_bool
4879     \peek_meaning:NTF [
4880       {
4881         \bool_set_eq:NN \l__enumext_wrap_label_viii_bool \l__enumext_wrap_label_opt_viii_bool
4882         \__enumext_start_item_viii:w
4883       }
4884       {
4885         \bool_set_true:N \l__enumext_wrap_label_viii_bool
4886         \legacy_if_set_true:n { @noitemarg }
4887         \__enumext_start_item_viii:w [ \l__enumext_label_viii_tl ] \ignorespaces
4888       }
4889   }
```

(*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_starred_item_exec:

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⟩].

```
4890 \cs_new_protected:Npn \__enumext_starred_item_viii:w
4891   {
4892     \bool_set_true:N \l__enumext_item_starred_viii_bool
4893     \bool_set_true:N \l__enumext_wrap_label_viii_bool
4894     \peek_meaning:NTF [
4895       { \__enumext_starred_item_viii_aux_i:w }
4896       { \__enumext_starred_item_viii_aux_ii:w }
4897   }
```

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.

```
4898 \cs_new_protected:Npn \__enumext_starred_item_viii_aux_i:w [#1]
4899   {
4900     \tl_clear:N \l__enumext_store_current_label_tl
4901     \tl_if_novalue:nF { #1 }
4902       {
4903         \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_tl
4904           {
4905             \tl_put_right:Ne \l__enumext_store_current_label_tl
4906               {
4907                 \l__enumext_store_keyans_item_opt_sep_tl
4908               }
4909             \tl_put_right:Ne \l__enumext_store_current_label_tl { #1 }
4910           }
4911         \tl_set:Ne \l__enumext_store_current_opt_arg_tl { #1 }
4912       }
4913     \__enumext_starred_item_viii_aux_ii:w
4914   }
4915 \cs_new_protected:Npn \__enumext_starred_item_viii_aux_ii:w
4916   {
4917     \legacy_if_set_true:n { @noitemarg }
4918     \__enumext_start_item_viii:w [ \l__enumext_label_viii_tl ] \ignorespaces
4919   }
```

The function __enumext_starred_item_exec: will be in charge of storing the current ⟨label⟩ for \item* followed by the [⟨content⟩] for \item*[⟨content⟩] if present in the *sequence* and *prop list* set by the save-ans key. In this same function the keys show-ans, show-pos and save-ref are implemented.

```
4920 \cs_new_protected:Nn \__enumext_starred_item_exec:
4921   {
4922     \tl_put_left:Ne \l__enumext_store_current_label_tl { \l__enumext_label_viii_tl }
4923     \__enumext_store_addto_prop:V \l__enumext_store_current_label_tl
4924     \__enumext_keyans_store_ref:
4925     \tl_put_left:Ne \l__enumext_store_current_label_tl { \item }
```

```
4926        \__enumext_keyans_addto_seq_link:
4927        \int_gincr:N \g__enumext_check_starred_cmd_int
4928        \bool_if:NT \l__enumext_show_answer_bool
4929          {
4930            \__enumext_print_keyans_box:NN \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
4931          }
4932        \bool_if:NT \l__enumext_show_position_bool
4933          {
4934            \tl_set:Ne \l__enumext_mark_answer_sym_tl
4935              {
4936                \group_begin:
4937                  \exp_not:N \normalfont
4938                  \exp_not:N \footnotesize [ \int_eval:n
4939                    {
4940                      \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
4941                    }
4942                  ]
4943                \group_end:
4944              }
4945            \__enumext_print_keyans_box:NN \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
4946          }
4947      }
```

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

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

```
4948  \cs_new_protected_nopar:Npn \__enumext_fake_make_label_viii:n #1
4949    {
4950      \legacy_if:nT { @noitemarg }
4951        {
4952          \legacy_if_set_false:n { @noitemarg }
4953          \legacy_if:nT { @nmbrlist }
4954            {
4955              \refstepcounter{enumXviii}
4956            }
4957        }
4958      \bool_if:NT \l__enumext_item_starred_viii_bool
4959        {
4960          \__enumext_starred_item_exec:
4961        }
4962      \makebox[ \l__enumext_labelwidth_viii_dim ][ \l__enumext_align_label_viii_str ]
4963        {
4964          \tl_use:N \l__enumext_label_font_style_viii_tl
4965          \bool_if:NTF \l__enumext_wrap_label_viii_bool
4966            {
4967              \__enumext_wrapper_label_viii:n {#1}
4968            }
4969            { #1 }
4970        }
4971      \skip_horizontal:N \l__enumext_labelsep_viii_dim \ignorespaces
4972    }
```

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

13.44.2 Real definition of \item in keyans*

__enumext_start_item_viii:w   The implementation at this is very similar to that of the enumext* environment.
__enumext_stop_item_viii:

```
4973  \cs_new_protected_nopar:Npn \__enumext_start_item_viii:w [#1]
4974    {
4975      \cs_set_eq:NN \__enumext_stop_item_tmp_viii: \__enumext_stop_item_viii:
4976      \hbox_set_to_wd:Nnw \l__enumext_item_text_viii_box
4977        {
4978          \l__enumext_joined_width_viii_dim
4979          + \l__enumext_labelwidth_viii_dim
4980          + \l__enumext_labelsep_viii_dim
4981        }
4982      \IfDocumentMetadataTF { } 
4983        {
4984          \bool_if:NF \l__enumext_footnotes_key_bool
4985            {
4986              \__enumext_renew_footnote:
4987            }
```

```
4988              }
4989          \__enumext_start_list_tag:n {keyans*}
4990          \__enumext_fake_make_label_viii:n {#1}
4991          \__enumext_stop_start_list_tag:
4992          \__enumext_minipage:w [ t ]{ \l__enumext_joined_width_viii_dim  }
4993            \dim_set_eq:NN \parindent \l__enumext_listparindent_viii_dim
4994            \skip_set_eq:NN \parskip \l__enumext_parsep_viii_skip
4995            \__enumext_unskip_unkern:
4996            \__enumext_unskip_unkern:
4997            \skip_horizontal:n { -\l__enumext_listparindent_viii_dim } \ignorespaces
4998            \tl_use:N \l__enumext_fake_item_indent_viii_tl
4999            \bool_if:NT \l__enumext_item_starred_viii_bool
5000              {
5001                \__enumext_keyans_show_item_opt:
5002              }
5003            \tl_use:N \l__enumext_after_list_args_viii_tl
5004      }
5005  \cs_new_protected_nopar:Nn \__enumext_stop_item_viii:
5006    {
5007        \__enumext_endminipage:
5008      \__enumext_stop_list_tag:n {keyans*}
5009      \hbox_set_end:
5010      \int_set:Nn \hbadness { 10000 }
5011      \box_use_drop:N \l__enumext_item_text_viii_box
5012      \IfDocumentMetadataTF { }
5013        {
5014          \bool_if:NF \l__enumext_footnotes_key_bool
5015            {
5016              \__enumext_print_footnote:
5017            }
5018        }
5019      \int_compare:nNnTF
5020        { \l__enumext_item_column_pos_viii_int } = { \l__enumext_columns_viii_int }
5021        {
5022          \par\noindent
5023          \int_zero:N \l__enumext_item_column_pos_viii_int
5024        }
5025        {
5026          \skip_horizontal:N \l__enumext_columns_sep_viii_dim
5027        }
5028    }
```

(*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.

```
5029  \cs_new_protected:Nn \__enumext_remove_extra_parsep_viii:
5030    {
5031      \int_compare:nNnT
5032        {
5033          \int_mod:nn
5034            { \g__enumext_item_count_all_viii_int }
5035            { \l__enumext_columns_viii_int }
5036        }
5037        =
5038        { 0 }
5039        {
5040          \para_end:
5041          \skip_vertical:n { -\l__enumext_itemsep_viii_skip }
5042          \skip_vertical:N \c_zero_skip
5043          \int_gzero:N \g__enumext_item_count_all_viii_int
5044        }
5045    }
```

(*End of definition for* __enumext_remove_extra_parsep_viii:.)

13.45   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.

```
5046  \NewDocumentCommand \getkeyans { m }
5047    {
```

```
5048        \exp_args:Ne \__enumext_getkeyans_aux:n
5049          { \tl_to_str:e { \text_expand:n {#1} } }
5050      }
```

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

```
5051  \cs_new_protected:Npn \__enumext_getkeyans_aux:n #1
5052    {
5053      \str_if_in:nnTF {#1} { : }
5054        {
5055          \use:e
5056            {
5057              \cs_set:Npn \exp_not:N \__enumext_tmp:w ##1 \c_colon_str ##2 \scan_stop:
5058                { {##1} {##2} }
5059            }
5060          \exp_after:wN \__enumext_getkeyans:nn \__enumext_tmp:w #1 \scan_stop:
5061        }
5062        { \msg_error:nnn { enumext } { missing-colon } {#1} }
5063    }
```

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*.

```
5064  \cs_new_protected:Npn \__enumext_getkeyans:nn #1 #2
5065    {
5066      \prop_if_exist:cTF { g__enumext_#1_prop }
5067        {
5068          \prop_item:cn { g__enumext_#1_prop }{#2}
5069        }
5070        {
5071          \msg_error:nnn { enumext } { undefined-storage-anskey } {#1}
5072        }
5073    }
```

(*End of definition for* `\getkeyans`, `\__enumext_getkeyans_aux:n`, *and* `\__enumext_getkeyans:nn`. *This function is documented on page 17.*)

13.46   The command `\printkeyans`

The `\printkeyans` command prints *"all stored content"* in the *sequence* defined by the `save-ans` key.
The first thing we will do is define a set of ⟨*filtered keys*⟩ with which we will control the options of the different nesting levels for the environment `enumext` and `enumext*` by storing their values in the list of tokens `\l__enumext_print_keyans_X_tl`.

The variable `\l__enumext_print_keyans_starred_tl` will have the default ⟨*keys*⟩ for `\printkeyans*` and will be set by `\setenumext[`⟨*print**⟩`]` and the variable `\l__enumext_print_keyans_vii_tl` will have the default keys for the environment `enumext*` nested within the *sequence* and will be set by `\setenumext[`⟨*print ,**⟩`]`, the rest of the variables will be for the environment `enumext` and will be set by `\setenumext[`⟨*print , level*⟩`]`.

```
5074  \keys_define:nn  { enumext / print }
5075    {
5076      print*  .code:n     = \keys_precompile:neN { enumext / enumext* }
5077                             { \__enumext_filter_save_key:n {#1} }
5078                           \l__enumext_print_keyans_starred_tl, % starred cmd
5079      print*  .initial:n = { nosep, label=\arabic*., columns=2, first=\small, font=\small },
5080      print-1 .code:n     = \keys_precompile:neN { enumext / level-1 }
5081                             { \__enumext_filter_save_key:n {#1} }
5082                           \l__enumext_print_keyans_i_tl,
5083      print-1 .initial:n = { nosep, label=\arabic*., columns=2, first=\small, font=\small },
5084      print-2 .code:n     = \keys_precompile:neN { enumext / level-2 }
5085                             { \__enumext_filter_save_key:n {#1} }
5086                           \l__enumext_print_keyans_ii_tl,
5087      print-2 .initial:n = { nosep, label=(\alph*), first=\small, font=\small },
5088      print-3 .code:n     = \keys_precompile:neN { enumext / level-3 }
5089                             { \__enumext_filter_save_key:n {#1} }
5090                           \l__enumext_print_keyans_iii_tl,
5091      print-3 .initial:n = { nosep, label=\roman*., first=\small, font=\small },
5092      print-4 .code:n     = \keys_precompile:neN { enumext / level-4 }
5093                             { \__enumext_filter_save_key:n {#1} }
5094                           \l__enumext_print_keyans_iv_tl,
5095      print-4 .initial:n = { nosep, label=\Alph*., first=\small, font=\small },
5096      print-* .code:n     = \keys_precompile:neN { enumext / enumext* }
```

```
5097                              { \__enumext_filter_save_key:n {#1} }
5098                              \l__enumext_print_keyans_vii_tl, % starred nested
5099      print-* .initial:n  = { nosep, label=\arabic*., first=\small, font=\small },
5100    }
```

💣 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.

```
5101  \NewDocumentCommand \printkeyans { s O{} m }
5102    {
5103      \group_begin:
5104        \tl_use:N \l__enumext_print_keyans_i_tl
5105        \tl_use:N \l__enumext_print_keyans_ii_tl
5106        \tl_use:N \l__enumext_print_keyans_iii_tl
5107        \tl_use:N \l__enumext_print_keyans_iv_tl
5108        \tl_use:N \l__enumext_print_keyans_vii_tl
5109        \__enumext_printkeyans:nnn { #1 } { #2 } { #3 }
5110      \group_end:
5111    }
```

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.

```
5112  \cs_new_protected:Npn \__enumext_printkeyans:nnn #1 #2 #3
5113    {
5114      \seq_if_exist:cTF { g__enumext_#3_seq }
5115        {
5116          \seq_if_empty:cF { g__enumext_#3_seq }
5117            {
```

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.

```
5118              \bool_if:nTF {#1}
5119                {
5120                  \seq_if_in:cnTF { g__enumext_#3_seq } { \end{enumext*} }
5121                    {
5122                      \msg_error:nnnn { enumext } { print-starred } {#3} { enumext* }
5123                    }
5124                    {
5125                      \tl_use:N \l__enumext_print_keyans_starred_tl
5126                      \bool_set_true:N \l__enumext_base_line_fix_bool
5127                      \bool_set_true:N \l__enumext_print_keyans_star_bool
5128                      \begin{enumext*}[#2]
5129                        \seq_map_inline:cn { g__enumext_#3_seq } { ##1 }
5130                      \end{enumext*}
5131                      \bool_set_false:N \l__enumext_base_line_fix_bool
5132                      \bool_set_false:N \l__enumext_print_keyans_star_bool
5133                    }
5134                }
```

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

```
5135                {
5136                  \begin{enumext}[#2]
5137                    \seq_map_inline:cn { g__enumext_#3_seq } { ##1 }
5138                  \end{enumext}
5139                }
5140            }
5141        }
5142        {
5143          \msg_error:nnn { enumext } { undefined-storage-anskey } {#3}
5144        }
5145    }
```

(*End of definition for* \printkeyans *and* __enumext_printkeyans:nnn. *This function is documented on page* 17.)

13.47   The command \setenumext

The command \setenumext will be in charge of managing the ⟨keys⟩ passed to all environments and to the \printkeyans command. We must take precautions with the enumext* environment and *"first level"* of the enumext environment so as not to capture ⟨keys⟩ that complicate us.

__enumext_filter_first_level:n
__enumext_filter_first_level_key:n
__enumext_filter_first_level_pair:nn

The function __enumext_filter_first_level:n will be in charge of filtering the ⟨keys⟩ passed to the environment enumext* and *"first level"* of the environment enumext.

```
5146  \cs_new:Npn \__enumext_filter_first_level:n #1
5147    {
5148      \use:e
5149        {
5150          \keyval_parse:NNn
5151            \__enumext_filter_first_level_key:n
5152            \__enumext_filter_first_level_pair:nn {#1}
5153        }
5154    }
```

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*.

```
5155  \cs_new:Npn \__enumext_filter_first_level_key:n #1
5156    {
5157      \str_case:nnF {#1}
5158        {
5159          { resume   } {}
5160          { resume*  } {}
5161        }
5162        { , { \exp_not:n {#1} } }
5163    }
```

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.

```
5164  \cs_new:Npn \__enumext_filter_first_level_pair:nn #1#2
5165    {
5166      \str_case:nnF {#1}
5167        {
5168          { series } {}
5169          { resume } {}
5170          { save-ans } {}
5171        }
5172        { , { \exp_not:n {#1} } = { \exp_not:n {#2} } }
5173    }
```

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

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

```
5174  \keys_define:nn { enumext / meta-families }
5175    {
5176      enumext-1 .code:n =
5177                  {
5178                    \keys_set:ne { enumext / level-1 }
5179                      {
5180                        \__enumext_filter_first_level:n {#1}
5181                      }
5182                  } ,
5183      enumext-2 .code:n = { \keys_set:nn { enumext / level-2 } {#1} } ,
5184      enumext-3 .code:n = { \keys_set:nn { enumext / level-3 } {#1} } ,
5185      enumext-4 .code:n = { \keys_set:nn { enumext / level-4 } {#1} } ,
5186      keyans    .code:n = { \keys_set:nn { enumext / keyans  } {#1} } ,
5187      enumext*  .code:n =
5188                  {
5189                    \keys_set:ne { enumext / enumext* }
5190                      {
5191                        \__enumext_filter_first_level:n {#1}
5192                      }
5193                  } ,
5194      keyans*   .code:n = { \keys_set:nn { enumext / keyans* } {#1} } ,
5195      print*    .code:n = { \keys_set:nn { enumext / print   } { print*  = {#1} } } ,
5196      print-1   .code:n = { \keys_set:nn { enumext / print   } { print-1 = {#1} } } ,
5197      print-2   .code:n = { \keys_set:nn { enumext / print   } { print-2 = {#1} } } ,
5198      print-3   .code:n = { \keys_set:nn { enumext / print   } { print-3 = {#1} } } ,
5199      print-4   .code:n = { \keys_set:nn { enumext / print   } { print-4 = {#1} } } ,
```

```
5200       print-*   .code:n = { \keys_set:nn { enumext / print   } { print-* = {#1} } } ,
5201       unknown   .code:n = { \msg_error:nn { enumext } { unknown-key-family } } ,
5202     }
```

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

```
5203  \seq_const_from_clist:Nn \c__enumext_all_families_seq
5204    {
5205       enumext-1, enumext-2, enumext-3, enumext-4, keyans, enumext*,
5206       keyans*, print-1, print-2, print-3, print-4, print-*, print*,
5207    }
```

\setenumext   Now we define the user command \setenumext.

__enumext_set_parse:n
__enumext_set_error:nn
```
5208  \NewDocumentCommand \setenumext { O{enumext,1} +m }
5209    {
5210       \seq_clear:N \l__enumext_setkey_tmpa_seq
5211       \seq_set_from_clist:Nn \l__enumext_setkey_tmpb_seq {#1}
5212       \int_set:Nn \l__enumext_setkey_tmpa_int
5213         {
5214            \seq_count:N \l__enumext_setkey_tmpb_seq
5215         }
5216       \int_compare:nNnTF { \l__enumext_setkey_tmpa_int } > { 1 }
5217         {
5218            \seq_pop_left:NN \l__enumext_setkey_tmpb_seq \l__enumext_setkey_tmpa_tl
5219            \seq_map_function:NN \l__enumext_setkey_tmpb_seq \__enumext_set_parse:n
5220            \seq_set_map_e:NNn \l__enumext_setkey_tmpa_seq \l__enumext_setkey_tmpa_seq
5221              {
5222                 \tl_use:N \l__enumext_setkey_tmpa_tl - ##1
5223              }
5224         }
5225         {
5226            \seq_put_right:Ne \l__enumext_setkey_tmpa_seq { \tl_trim_spaces:n {#1} }
5227         }
5228       \seq_if_empty:NTF \l__enumext_setkey_tmpa_seq
5229         { \seq_map_inline:Nn \c__enumext_all_families_seq }
5230         { \seq_map_inline:Nn \l__enumext_setkey_tmpa_seq }
5231         {
5232            \keys_set:nn { enumext / meta-families } { ##1 = {#2} }
5233         }
5234    }
```

Internal functions used by the \setenumext command.

```
5235  \cs_new_protected:Npn \__enumext_set_parse:n #1
5236    {
5237       \tl_set:Ne \l__enumext_setkey_tmpb_tl { \tl_trim_spaces:n {#1} }
5238       \clist_map_inline:nn { 0, 1, 2, 3, 4, * } % <- max level
5239         { \tl_remove_all:Nn \l__enumext_setkey_tmpb_tl {##1} }
5240       \tl_if_empty:NTF \l__enumext_setkey_tmpb_tl
5241         {
5242            \seq_put_right:Ne \l__enumext_setkey_tmpa_seq
5243              { \tl_trim_spaces:n {#1} }
5244         }
5245         { \__enumext_set_error:nn {#1} { } }
5246    }
5247  \cs_new_protected:Npn \__enumext_set_error:nn #1 #2
5248    { \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.*)

13.48   The command \setenumextmeta

The command \setenumextmeta will be responsible for adding new *"meta-keys"* for the enumext and enumext* environments. The implementation code was given by Jonathan P. Spratte (@Skillmon) answer in Add .meta key to existing keys (l3keys).

\setenumextmeta   First we will create a prop list \c__enumext_meta_paths_prop to handle the *optional argument*.

\c__enumext_meta_paths_prop
__enumext_add_meta_key:nnn
__enumext_def_meta_key:nnn
__enumext_def_meta_key:Vnn
```
5249  \prop_const_from_keyval:Nn \c__enumext_meta_paths_prop
5250    {
5251       {enumext,1} = level-1,
5252       {enumext,2} = level-2,
5253       {enumext,3} = level-3,
5254       {enumext,4} = level-4,
```

```
5255      {enumext*}  = enumext*
5256    }
```

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

```
5257  \NewDocumentCommand \setenumextmeta { s O{enumext,1} m +m }
5258    {
5259      \str_if_eq:eeTF { \tl_trim_spaces:n {#3} } { unknown }
5260        { \msg_error:nn { enumext } { prohibited-unknown } }
5261        {
5262          \bool_if:nTF {#1}
5263            {
5264              \int_step_inline:nn { 4 }
5265                { \__enumext_add_meta_key:nnn { enumext, ##1 } {#3} {#4} }
5266              \__enumext_add_meta_key:nnn { enumext* } {#3} {#4}
5267            }
5268            { \__enumext_add_meta_key:nnn {#2} {#3} {#4} }
5269        }
5270    }
```

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

```
5271  \cs_new_protected:Npn \__enumext_add_meta_key:nnn #1
5272    {
5273      \tl_set:Nn \l__enumext_meta_path_tl {#1}
5274      \tl_replace_all:Nnn \l__enumext_meta_path_tl { ~ } {}
5275      \prop_get:NVNTF
5276        \c__enumext_meta_paths_prop \l__enumext_meta_path_tl \l__enumext_meta_path_tl
5277        { \__enumext_def_meta_key:Vnn \l__enumext_meta_path_tl }
5278        {
5279          \msg_error:nnn { enumext } { unknown-set } {#1}
5280          \use_none:nn
5281        }
5282    }
5283  \cs_new_protected:Npn \__enumext_def_meta_key:nnn #1#2#3
5284    {
5285      \bool_lazy_or:nnTF
5286        { \keys_if_exist_p:nn { enumext / #1 } {#2} }
5287        { \keys_if_exist_p:nn { enumext / enumext* } {#2} }
5288        { \msg_error:nnn { enumext } { already-defined } {#2} }
5289        {
5290          \keys_define:nn { enumext / #1 }
5291            {
5292              #2 .meta:n = {#3},
5293              #2 .value_forbidden:n = true
5294            }
5295        }
5296    }
5297  \cs_generate_variant:Nn \__enumext_def_meta_key:nnn { V }
```

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

13.49 The command \foreachkeyans

The command \foreachkeyans will execute a *loop* over the *prop list* and return its contents. The implementation code is adapted from the answer provided by Enrico Gregorio (@egreg) in Expand a `.cs` defined by key inside the function.

\foreachkeyans
__enumext_parse_foreach_keys:nn
__enumext_parse_foreach_keys:n
__enumext_foreach_keyans:nn
__enumext_foreach_add_body:n

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

```
5298  \keys_define:nn { enumext / foreach }
5299    {
5300      before  .tl_set:N  = \l__enumext_foreach_before_tl,
5301      before  .value_required:n = true,
5302      after   .tl_set:N  = \l__enumext_foreach_after_tl,
5303      after   .value_required:n = true,
5304      start   .int_set:N = \l__enumext_foreach_start_int,
5305      start   .value_required:n = true,
5306      stop    .int_set:N = \l__enumext_foreach_stop_int,
5307      stop    .value_required:n = true,
5308      step    .int_set:N = \l__enumext_foreach_step_int,
5309      step    .value_required:n = true,
5310      wrapper .cs_set_protected:Np = \__enumext_foreach_wrapper:n #1,
5311      wrapper .value_required:n = true,
```

```
5312      sep     .tl_set:N  = \l__enumext_foreach_sep_tl,
5313      sep     .value_required:n = true,
5314      unknown .code:n     = { \__enumext_parse_foreach_keys:n {#1} }
5315    }
5316  \keys_precompile:nnN { enumext / foreach }
5317    {
5318      before={},after={},start=1,step=1,stop=0,wrapper=#1,sep={; }
5319    }
5320    \g__enumext_foreach_default_keys_tl
```

Functions for handling unknown ⟨keys⟩.

```
5321  \cs_new_protected:Npn \__enumext_parse_foreach_keys:nn #1#2
5322    {
5323      \tl_if_blank:nTF {#2}
5324        {
5325          \msg_error:nnn { enumext } { for-key-unknown } {#1}
5326        }
5327        {
5328          \msg_error:nnnn { enumext } { for-key-value-unknown } {#1} {#2}
5329        }
5330    }
5331  \cs_new_protected:Npn \__enumext_parse_foreach_keys:n #1
5332    {
5333      \exp_args:NV \__enumext_parse_foreach_keys:nn \l_keys_key_str {#1}
5334    }
```

We create the command.

```
5335  \NewDocumentCommand \foreachkeyans { +O{} m }
5336    {
5337      \__enumext_foreach_keyans:nn {#1} {#2}
5338    }
```

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

```
5339  \cs_new_protected:Npn \__enumext_foreach_keyans:nn #1 #2
5340    {
5341      \tl_use:N \g__enumext_foreach_default_keys_tl
5342      \keys_set:nn { enumext / foreach } {#1}
5343      \tl_set:Nn \l__enumext_foreach_name_prop_tl {#2}
5344      \prop_if_exist:cF { g__enumext_#2_prop }
5345        {
5346          \msg_error:nnn { enumext } { undefined-storage-anskey } {#2}
5347        }
5348      \int_compare:nNnT { \l__enumext_foreach_stop_int } = { 0 }
5349        {
5350          \int_set:Nn \l__enumext_foreach_stop_int
5351            { \prop_count:c { g__enumext_#2_prop } }
5352        }
5353      \seq_clear:N \l__enumext_foreach_print_seq
5354      \int_step_function:nnnN
5355        { \l__enumext_foreach_start_int }
5356        { \l__enumext_foreach_step_int }
5357        { \l__enumext_foreach_stop_int }
5358        \__enumext_foreach_add_body:n
5359        \seq_use:NV \l__enumext_foreach_print_seq \l__enumext_foreach_sep_tl
5360    }
5361  \cs_new_protected:Npn \__enumext_foreach_add_body:n #1
5362    {
5363      \seq_put_right:Ne \l__enumext_foreach_print_seq
5364        {
5365          \exp_not:V \l__enumext_foreach_before_tl
5366          \__enumext_foreach_wrapper:n
5367            {
5368              \prop_item:cn { g__enumext_ \l__enumext_foreach_name_prop_tl _prop }{#1}
5369            }
5370          \exp_not:V \l__enumext_foreach_after_tl
5371        }
5372    }
```

(*End of definition for* \foreachkeyans *and others. This function is documented on page* 17.)

13.50 Messages

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

```
5373 \msg_new:nnn { enumext } { package-load }
5374   {
5375     The ~ '#1' ~ package ~ is ~ already ~ loaded.
5376   }
5377 \msg_new:nnn { enumext } { package-not-load }
5378   {
5379     The ~ '#1' ~ package ~ will ~ be ~ loaded ~ as ~ a ~ dependency.
5380   }
5381 \msg_new:nnn { enumext } { package-load-foot }
5382   {
5383     The ~ '#1' ~ package ~ is ~ loaded ~ with ~ the ~ option ~ '#2'.
5384   }
```

Message used in the creation of counters by enumext package.

```
5385 \msg_new:nnn { enumext } { counters }
5386   {
5387     The ~ counter ~ '#1' ~ is ~ already ~ defined ~ by ~ some ~ \\
5388     package ~ or ~ macro, ~ it ~ cannot ~ be ~ continued.
5389   }
```

Message used by align and mark-pos keys.

```
5390 \msg_new:nnn { enumext } { unknown-choice }
5391   {
5392     The ~ value ~ '#3' ~ for ~ '#1' ~ key ~ is ~ invalid ~ use ~ ('#2').
5393   }
```

Message used by reserved anskey* environment by enumext package.

```
5394 \msg_new:nnnn { enumext } { anskey-env-error }
5395   {
5396     The ~ '#1' ~ environment ~is ~ reserved ~ by ~\\
5397     'enumext' ~ package, ~ It~ is~ already~ defined.
5398   }
5399   {
5400     The ~ anskey* ~ environment ~ is ~ defined ~ internally ~
5401     for ~ the ~ 'save-ans' ~ key.\\
5402   }
```

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

```
5403 \msg_new:nnn { enumext } { store-prop }
5404   {
5405     * ~ Package ~ enumext: ~ Creating ~
5406     \c_backslash_str g__enumext_#1_prop ~ \msg_line_context:.
5407   }
5408 \msg_new:nnn { enumext } { store-seq }
5409   {
5410     * ~ Package ~ enumext: ~ Creating ~
5411     \c_backslash_str g__enumext_#1_seq ~ \msg_line_context:.
5412   }
5413 \msg_new:nnn { enumext } { store-int }
5414   {
5415     * ~ Package ~ enumext: ~ Creating ~
5416     \c_backslash_str g__enumext_resume_#1_int ~ \msg_line_context:.
5417   }
5418 \msg_new:nnn { enumext } { prop-seq-int-hook }
5419   {
5420     * ~ Package ~ enumext: ~ Elements ~ in ~
5421     \c_backslash_str g__enumext_#1_prop ~ = ~ #2.\\
5422     * ~ Package ~ enumext: ~ Elements ~ in ~
5423     \c_backslash_str g__enumext_#1_seq ~ = ~ #3.\\
5424     * ~ Package ~ enumext: ~ Value ~ off ~
5425     \c_backslash_str g__enumext_resume_#1_int ~ = ~ #4.
5426   }
5427 \msg_new:nnn { enumext } { item-answer-hook }
5428   {
5429     * ~ Package ~ enumext: ~ Value ~ off ~
5430     \c_backslash_str g__enumext_item_number_int ~ = ~ #1.\\
5431     * ~ Package ~ enumext: ~ Value ~ off ~
5432     \c_backslash_str g__enumext_item_anskey_int ~ = ~ #2.\\
5433     * ~ Package ~ enumext: ~ Difference ~ item_number_int ~ - ~ item_anskey_int ~ = ~ #3.
5434   }
```

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

```
5435  \msg_new:nnn { enumext } { invalid-key }
5436    {
5437      The ~ key ~ '#1' ~ is ~ not ~ know ~ the ~ level ~ #2.
5438    }
5439  \msg_new:nnn { enumext } { unknown-key-family }
5440    {
5441      Unknown~key~family~`\l_keys_key_str'~for~enumext.
5442    }
```

Messages used in length calculation.

```
5443  \msg_new:nnn { enumext } { width-negative }
5444    {
5445      Ignoring ~ negative ~ value ~ '#1=#2' ~ \msg_line_context:.\\
5446      The ~ key ~ '#1'~ accepts ~ values  ~ >= ~ 0pt.
5447    }
5448  \msg_new:nnn { enumext } { width-zero }
5449    {
5450      Invalid ~ '#1=#2' ~ \msg_line_context:.\\
5451      The ~ key ~ '#1'~ accepts ~ values  ~ > ~ 0pt.
5452    }
```

Messages used by show-length key in enumext.

```
5453  \msg_new:nnn { enumext } { list-lengths }
5454    {
5455      **** ~ Lengths ~ used ~ by ~ 'enumext' ~ level ~ '#2' ~ \msg_line_context:~\c_space_tl ****\\
5456      \__enumext_show_length:nnn { dim  } { labelsep     } {#1}
5457      \__enumext_show_length:nnn { dim  } { labelwidth   } {#1}
5458      \__enumext_show_length:nnn { dim  } { itemindent   } {#1}
5459      \__enumext_show_length:nnn { dim  } { leftmargin   } {#1}
5460      \__enumext_show_length:nnn { dim  } { rightmargin  } {#1}
5461      \__enumext_show_length:nnn { dim  } { listparindent } {#1}
5462      \__enumext_show_length:nnn { skip } { topsep    } {#1}
5463      \__enumext_show_length:nnn { skip } { parsep    } {#1}
5464      \__enumext_show_length:nnn { skip } { partopsep } {#1}
5465      \__enumext_show_length:nnn { skip } { itemsep   } {#1}
5466      *************************************************
5467    }
```

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

```
5468  \msg_new:nnn { enumext } { list-lengths-not-nested }
5469    {
5470      **** ~ Lengths ~ used ~ by ~ '#2' ~ environment ~ \msg_line_context:~\c_space_tl ****\\
5471      \__enumext_show_length:nnn { dim  } { labelsep     } {#1}
5472      \__enumext_show_length:nnn { dim  } { labelwidth   } {#1}
5473      \__enumext_show_length:nnn { dim  } { itemindent   } {#1}
5474      \__enumext_show_length:nnn { dim  } { leftmargin   } {#1}
5475      \__enumext_show_length:nnn { dim  } { rightmargin  } {#1}
5476      \__enumext_show_length:nnn { dim  } { listparindent } {#1}
5477      \__enumext_show_length:nnn { skip } { topsep    } {#1}
5478      \__enumext_show_length:nnn { skip } { parsep    } {#1}
5479      \__enumext_show_length:nnn { skip } { partopsep } {#1}
5480      \__enumext_show_length:nnn { skip } { itemsep   } {#1}
5481      *************************************************
5482    }
```

Messages used by ref key.

```
5483  \msg_new:nnn { enumext } { key-ref-empty }
5484    {
5485      Key ~ 'ref' ~ need ~ a ~ value ~ in ~ '#1'~ \msg_line_context:.
5486    }
```

Messages used by save-ans key.

```
5487  \msg_new:nnn { enumext } { save-ans-empty }
5488    {
5489      Key ~ 'save-ans' ~ need ~ a ~ value ~ in ~ '#1'~ \msg_line_context:.
5490    }
5491  \msg_new:nnn { enumext } { save-ans-log }
5492    {
5493      * ~ Package ~ enumext: ~ Start ~ #1\c_space_tl with ~ save-ans=#2 ~ \msg_line_context:.
5494    }
5495  \msg_new:nnn { enumext } { save-ans-log-hook }
5496    {
```

```
5497        * ~ Package ~ enumext: ~ Stop ~ #1\c_space_tl with ~ save-ans=#2 ~ \msg_line_context:.
5498      }
5499  \msg_new:nnn { enumext } { save-ans-hook }
5500    {
5501      Stop ~ storing ~ for ~ 'save-ans=#1' ~ \msg_line_context:.
5502    }
```

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

```
5503  \msg_new:nnn { enumext } { need-save-ans }
5504    {
5505      Key ~ '#1'~ works ~ only ~ with ~ the ~ 'save-ans' ~ key ~ in ~ '#2'~ \msg_line_context:.
5506    }
5507  \msg_new:nnn { enumext } { items-same-answer }
5508    {
5509      ****************************************\\
5510      * ~ Package ~ enumext: ~ Checking ~ answers ~ in ~ '#1' ~
5511      for ~ \c_left_brace_str #2 \c_right_brace_str\\
5512      * ~ started ~ #3 ~ and ~ close ~ \msg_line_context: : ~
5513      'OK', ~ all ~ items ~ with ~ answer.\\
5514      ****************************************
5515    }
5516  \msg_new:nnn { enumext } { item-greater-answer }
5517    {
5518      Checking ~ answers ~ in ~ '#1' ~ for ~ \c_left_brace_str #2 \c_right_brace_str\\
5519      started ~ #3 ~ and ~ close ~ \msg_line_context: : ~'NOT ~ OK'\\
5520      Items ~ > ~ Answers.
5521    }
5522  \msg_new:nnn { enumext } { item-less-answer }
5523    {
5524      Checking ~ answers ~ in ~ '#1' ~ for ~ \c_left_brace_str #2 \c_right_brace_str\\
5525      started ~ #3 ~ and ~ close ~ \msg_line_context: : ~'NOT ~ OK'\\
5526      Items ~ < ~ Answers.
5527    }
```

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

```
5528  \msg_new:nnn { enumext } { missing-starred }
5529    {
5530      Missing ~ '\c_backslash_str #1*' ~ #2.
5531    }
5532  \msg_new:nnn { enumext } { many-starred }
5533    {
5534      Many ~ '\c_backslash_str #1*' ~ #2.
5535    }
```

Messages used by \printkeyans* command.

```
5536  \msg_new:nnn { enumext } { print-starred }
5537    {
5538      \c_backslash_str printkeyans*:~ The ~ sequence ~ '#1' ~ already ~ contains ~
5539      #2 ~ environment ~  \msg_line_context:.
5540    }
```

Message for the nesting depth of the environment enumext.

```
5541  \msg_new:nnn { enumext } { list-too-deep }
5542    {
5543      Too ~ deep ~ nesting  ~ for ~ 'enumext' ~ \msg_line_context:.~ \\
5544      The ~ maximum  ~ level  ~ of  ~ nesting  ~ is ~ 4.
5545    }
```

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

```
5546  \msg_new:nnn { enumext } { anskey-unnumber-item }
5547    {
5548      Can't ~ store ~ with ~ a ~ unnumbered ~ \c_backslash_str item ~ \msg_line_context:.
5549    }
5550  \msg_new:nnn { enumext } { anskey-already-stored }
5551    {
5552      Content ~ already ~ stored ~ for ~ this ~ \c_backslash_str item ~ \msg_line_context:.
5553    }
5554  \msg_new:nnn { enumext } { anskey-empty-arg }
5555    {
5556      Can't ~ store ~ empty ~ content ~ \msg_line_context:.
5557    }
5558  \msg_new:nnn { enumext } { anskey-wrong-place }
5559    {
```

```
5560        Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:.~ \\
5561        '\c_backslash_str #1' ~ works ~ in ~ the ~ environment ~ '#2'.
5562      }
5563 \msg_new:nnn { enumext } { anskey-nested }
5564      {
5565        The ~ command ~ \c_backslash_str anskey~ can't ~ be ~ nested ~ \msg_line_context:.
5566      }
5567 \msg_new:nnn { enumext } { anskey-math-mode }
5568      {
5569        #1 ~ can't ~ work ~ in ~ math ~ mode ~ \msg_line_context:.
5570      }
5571 \msg_new:nnn { enumext } { anskey-env-wrong }
5572      {
5573        The ~ environment ~ anskey* ~ cannot ~ use ~ in ~ '#1' ~ \msg_line_context:.
5574      }
5575 \msg_new:nnn { enumext } { anspic-wrong-place }
5576      {
5577        Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:.~ \\
5578        '\c_backslash_str #1' ~ works ~ in ~ the ~ environment ~ '#2'.
5579      }
5580 \msg_new:nnn { enumext } { command-wrong-place }
5581      {
5582        Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:.~ \\
5583        '\c_backslash_str #1' ~ works ~ outside ~ the ~ environment ~ '#2'.
5584      }
5585 \msg_new:nnnn { enumext } { anskey-env-key-unknown }
5586      {
5587        The ~ key ~ '#1' ~ is ~ unknown ~ by ~ environment~
5588        'anskey*' ~ and ~ is ~ being ~ ignored.
5589      }
5590      {
5591        The ~ environment ~ 'anskey*' ~ does ~ not ~ have ~ a ~ key ~ called ~'#1'.\\
5592        Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5593      }
5594 \msg_new:nnnn { enumext } { anskey-env-key-value-unknown }
5595      {
5596        The ~ key ~ '#1=#2' ~ is ~ unknown ~ by ~ environment ~
5597        'anskey*' ~ and ~ is ~ being ~ ignored.
5598      }
5599      {
5600        The ~ environment ~ 'anskey*' ~ does ~ not ~ have ~ a ~ key ~ called ~'#1'.\\
5601        Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5602      }
5603 \msg_new:nnnn { enumext } { anskey-cmd-key-unknown }
5604      { The ~ key ~'#1'~ is ~ unknown ~ by ~ '\c_backslash_str anskey' ~ and ~ is ~ being ~ ignored.}
5605      {
5606        The ~ command ~'\c_backslash_str anskey' ~ does ~ not ~ have ~ a ~ key ~ called ~'#1'.\\
5607        Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5608      }
5609 \msg_new:nnnn { enumext } { anskey-cmd-key-value-unknown }
5610      { The ~ key ~ '#1=#2' ~ is ~ unknown ~ by ~ '\c_backslash_str anskey' ~ and ~ is ~ being ~ igno
5611      {
5612        The ~ command ~ '\c_backslash_str anskey' ~ does ~ not ~ have ~ a ~ key ~ called ~'#1'.\\
5613        Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5614      }
```

Messages used by keyans, keyans* and keyanspic environment.

```
5615 \msg_new:nnn { enumext } { keyans-nested }
5616      {
5617        The ~ environment ~ 'keyans' ~ can't ~ be  ~ nested  ~ \msg_line_context:.
5618      }
5619 \msg_new:nnn { enumext } { keyans-wrong-level }
5620      {
5621        Wrong ~ level ~ position ~ for ~ 'keyans' ~ \msg_line_context:.~ \\
5622        The ~ environment ~ 'keyans' ~ can ~ only ~ be ~ in ~ the ~ first ~ level.
5623      }
5624 \msg_new:nnn { enumext } { wrong-place }
5625      {
5626        Wrong ~ place ~ for ~ '#1' ~ environment ~\msg_line_context:.~ \\
5627        '#1' ~ is ~ only ~ found ~ with ~ '#2' ~  in  ~  'enumext.
5628      }
5629 \msg_new:nnn { enumext } { keyanspic-nested }
```

```
5630    {
5631      The ~ environment ~ 'keyanspic' ~ can't ~ be  ~ nested~ \msg_line_context:.~.
5632    }
5633  \msg_new:nnn { enumext } { keyanspic-wrong-level }
5634    {
5635      Wrong ~ level ~ position ~ for ~ 'keyanspic' ~ \msg_line_context:.~ \\
5636      The ~ environment ~ 'keyans' ~ can ~ only ~ be ~ in ~ the ~ first ~ level.
5637    }
5638  \msg_new:nnn { enumext } { keyanspic-item-cmd }
5639    {
5640      Can't ~ use  ~ \c_backslash_str item ~ in ~ keyanspic ~ \msg_line_context:.
5641    }
5642  \msg_new:nnnn { enumext } { keyans-unknown-key }
5643    {
5644      The ~ key ~ '#1' ~ is ~ unknown ~ by ~ environment~
5645      '\l__enumext_envir_name_tl' ~ and ~ is ~ being ~ ignored.
5646    }
5647    {
5648      The ~ environment ~ '\l__enumext_envir_name_tl' ~ does ~ not
5649      ~ have ~ a ~ key ~ called ~'#1'.\\
5650      Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5651    }
5652  \msg_new:nnnn { enumext } { keyans-unknown-key-value }
5653    {
5654      The ~ key ~ '#1=#2' ~ is ~ unknown ~ by ~ environment ~
5655      '\l__enumext_envir_name_tl' ~ and ~ is ~ being ~ ignored.
5656    }
5657    {
5658      The ~ environment ~ '\l__enumext_envir_name_tl' ~ does ~ not
5659      ~ have ~ a ~ key ~ called ~'#1'.\\
5660      Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5661    }
```

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

```
5662  \msg_new:nnnn { enumext } { starred-unknown-key }
5663    {
5664      The ~ key ~ '#1' ~ is ~ unknown ~ by ~ environment~
5665      '\l__enumext_envir_name_tl' ~ and ~ is ~ being ~ ignored.
5666    }
5667    {
5668      The ~ environment ~ '\l__enumext_envir_name_tl' ~ does ~ not
5669      ~ have ~ a ~ key ~ called ~'#1'.\\
5670      Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5671    }
5672  \msg_new:nnnn { enumext } { starred-unknown-key-value }
5673    {
5674      The ~ key ~ '#1=#2' ~ is ~ unknown ~ by ~ environment ~
5675      '\l__enumext_envir_name_tl' ~ and ~ is ~ being ~ ignored.
5676    }
5677    {
5678      The ~ environment ~ '\l__enumext_envir_name_tl' ~ does ~ not
5679      ~ have ~ a ~ key ~ called ~'#1'.\\
5680      Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5681    }
```

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

```
5682  \msg_new:nnnn { enumext } { standar-unknown-key }
5683    {
5684      The ~ key ~ '#1' ~ is ~ unknown ~ by ~ environment ~ '\l__enumext_envir_name_tl' \c_space_tl
5685      ~ on ~ level ~ \int_use:N \l__enumext_level_int \c_space_tl and ~ is ~ being ~ ignored.
5686    }
5687    {
5688      The ~ environment ~ '\l__enumext_envir_name_tl' ~ does ~ not
5689      ~ have ~ a ~ key ~ called ~'#1' ~ on ~ level ~ \int_use:N \l__enumext_level_int.\\
5690      Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5691    }
5692  \msg_new:nnnn { enumext } { standar-unknown-key-value }
5693    {
5694      The ~ key ~ '#1=#2' ~ is ~ unknown ~ by ~ environment ~ '\l__enumext_envir_name_tl' \c_space_
5695      ~ on ~ level ~ \int_use:N \l__enumext_level_int \c_space_tl and ~ is ~ being ~ ignored.
5696    }
5697    {
```

```
5698        The ~ environment ~ '\l__enumext_envir_name_tl' ~ does ~ not
5699        ~ have ~ a ~ key ~ called ~'#1' ~ on ~ level ~ \int_use:N \l__enumext_level_int.\\
5700        Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5701      }
```

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

```
5702  \msg_new:nnnn { enumext } { for-key-unknown }
5703    { The~key~'#1'~is~unknown~by~'\c_backslash_str foreachkeyans'~and~is~being~ignored.}
5704    {
5705      The~command~'\c_backslash_str foreachkeyans'~does~not~have~a~key~called~'#1'.\\
5706      Check~that~you~have~spelled~the~key~name~correctly.
5707    }
5708  \msg_new:nnnn { enumext } { for-key-value-unknown }
5709    { The~key~'#1=#2'~is~unknown~by~'\c_backslash_str foreachkeyans'~and~is~being~ignored. }
5710    {
5711      The~command~'\c_backslash_str foreachkeyans'~does~not~have~a~key~called~'#1'.\\
5712      Check~that~you~have~spelled~the~key~name~correctly.
5713    }
```

Messages used by \getkeyans command.

```
5714  \msg_new:nnn { enumext } { undefined-storage-anskey }
5715    {
5716      Storage ~ named ~ '#1' ~ is ~ not ~ defined ~ \msg_line_context:.
5717    }
```

Messages used by \miniright command.

```
5718  \msg_new:nnn { enumext } { missing-miniright }
5719    {
5720      Missing ~ '\c_backslash_str miniright' ~ in ~ \msg_line_context:.\\
5721      The ~ key ~ 'mini-env' ~ need ~ '\c_backslash_str miniright'.
5722    }
5723  \msg_new:nnn { enumext } { wrong-miniright-place }
5724    {
5725      Wrong ~ place ~ for ~ '\c_backslash_str miniright' ~ \msg_line_context:.~ \\
5726      Works ~ in ~ 'enumext' ~ and ~ 'keyans' ~ with ~ key ~ 'mini-env'.
5727    }
5728  \msg_new:nnn { enumext } { wrong-miniright-use }
5729    {
5730      Wrong ~ use ~ for ~ '\c_backslash_str miniright' ~ \msg_line_context:.~ \\
5731      '\c_backslash_str miniright' ~ need ~ a ~ key ~ 'mini-env'.
5732    }
5733  \msg_new:nnn { enumext } { wrong-miniright-starred }
5734    {
5735      Can't ~ use  ~ \c_backslash_str miniright ~ in ~ starred ~ environments ~ \msg_line_context:.
5736    }
5737  \msg_new:nnn { enumext } { many-miniright-used }
5738    {
5739      Can't ~ use  ~ \c_backslash_str miniright ~ more ~ than ~ once ~  \msg_line_context:.
5740    }
```

Messages used by \setenumextmeta command.

```
5741  \msg_new:nnn { enumext } { unknown-set }
5742    {
5743      Argument ~ [#1] ~ is ~ unknown ~ by ~  \c_backslash_str setenumextmeta ~ \msg_line_context:.
5744    }
5745  \msg_new:nnn { enumext } { already-defined }
5746    {
5747      The ~ key ~ '#1' ~ is ~ already ~ defined ~ \msg_line_context:.
5748    }
5749  \msg_new:nnn { enumext } { prohibited-unknown }
5750    {
5751      The ~ name ~ 'unknown' ~ can't ~ be ~ chosen~ for ~ a ~ meta ~ key ~ \msg_line_context:.
5752    }
```

Messages used by enumext* and keyans* environments.

```
5753  \msg_new:nnn { enumext } { nested }
5754    {
5755      The ~ environment ~ \l__enumext_envir_name_tl \c_space_tl can't ~ be ~ nested ~ \msg_line_con
5756    }
5757  \msg_new:nnn { enumext } { nested-horizontal }
5758    {
5759      The ~ environment ~ \l__enumext_envir_name_tl \c_space_tl can't ~ be ~ nested ~ in ~ '#1' ~ '
5760    }
```

```
5761  \msg_new:nnn { enumext } { item-joined }
5762    {
5763      Items ~ joined ~ (#1) ~ > ~ #2  ~ columns ~\msg_line_context:.
5764    }
5765  \msg_new:nnn { enumext } { item-joined-columns }
5766    {
5767      Not ~ space ~ to ~ join ~ items ~ (#1) ~ > ~ #2 ~\msg_line_context:.
5768    }
```

13.51   Finish package

Finish package implementation.

```
5769  \file_input_stop:
5770  ⟨/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.

K