enumext

ENUMERATE EXERCISE SHEETS

V1.0   2024-05-22[*]

©2024 by Pablo González[†]

Abstract

This package provides *"enumerated list"* environments for creating *"simple exercise sheets"* along with *"multiple choice questions"*, storing the ⟨*answers*⟩ to these in memory using the `multicol` package 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] package, need to have a modern TeX distribution such as TeX Live or MiKTeX. It has been tested with the standard classes provided by LaTeX: `book`, `report`, `article` and `letter` on `10pt`, `11pt` and `12pt`.

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

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

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

4. Question with image and label below:

(A) (B) (C)

(D) (E)
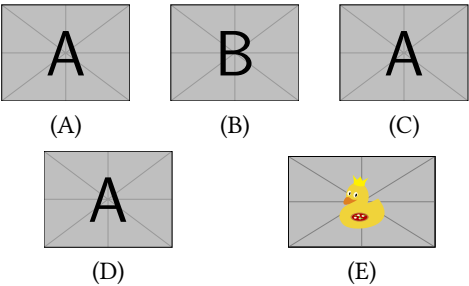
5. Question with image on left side:

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

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

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

4. (B)
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[5], 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[5] package adds the `\labelindent` parameter to solve some of these problems. A simplified representation of this in the figure 1.
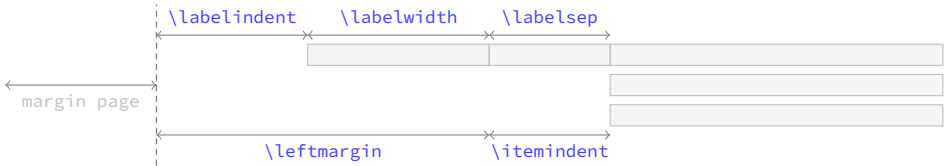

Figure 1: Representation of horizontal lengths in enumitem.

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.
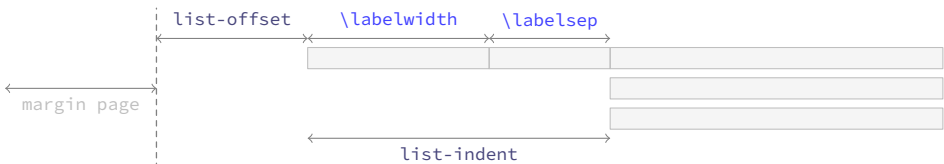

Figure 2: Representation of horizontal lengths concept in enumext.

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 in enumext, enumext*, keyans, keyans* and keyanspic environments, `\anskey`, `\item*` and `\anspic*` commands to ⟨*stored content*⟩, `\getkeyans` command to get the 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 missing error and abort the load.

1.3.2 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.
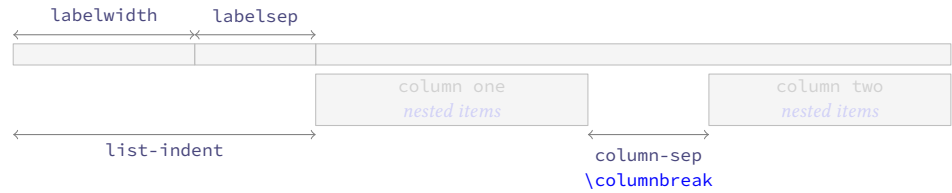

Figure 4: Representation of the two column output for a nested level in `enumext` environment.

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. The environment is available for all nesting levels, and can can together with the `mini-env` key. If you need to force a start a new column `\columnbreak` must be used (see §3.5).

💣 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.

1.3.3 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 (left and right) is always used with *"aligned on top"* [t], the `minipage` environment on the *"right side"* always starts with `\centering`. It can be used at all nesting levels 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 §3.6).

1.3.4 The `\label` and `\ref` system

This package provides a user interface like the `enumitem`[5] package to customize the references which is activated by the `ref` key (§3.1), 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` (§4.2) when the key `save-ans` (§4.1) is active.

💣 The implementation of `\label` and `\ref` together with the `save-ref` key are compatible with the `hyperref`[7] package.

1.3.5 Support for `\footnote`

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

The best way to solve this is to use Jean-François Burnol `footnotehyper`[8] 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*}
```

2 The environment `enumext`

enumext
enumext*

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

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

The `enumext` is an *"enumerated list"* 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

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.

```
\begin{enumext}
  \item This text is in the first level.
    \begin{enumext}
      \item This text is in the second level.
        \begin{enumext}
          \item This text is in the third level.
            \begin{enumext}
              \item This text is in the fourth level.
            \end{enumext}
        \end{enumext}
    \end{enumext}
  \item[X] This text is in the first level.
  \item* This text is in the first level.
\end{enumext}
```

2.1 The `\item*` in enumext

`\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 value set by the `labelsep` key and can be ⟨*offset*⟩ using the second optional argument. The default values for ⟨*symbol*⟩ and ⟨*offset*⟩ are `$\star$` '⋆' and the value set by `labelsep` key.

The *starred version* '*' 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` environment is NOT the same as in the `keyans` environment.

2.1.1 Keys for `\item*` in enumext

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

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

`item-pos*` = {⟨*rigid length* | *dim expression*⟩}   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*.

3 The command `\setenumext`

`\setenumext`

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

The command `\setenumext` sets the ⟨*keys*⟩ on a global basis for environment `enumext`, the `\printkeyans` command and the `keyans` environment. 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 arguments 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.

💣* 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` and `keyanspic` 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.

3.1 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 first level are `\arabic*.`, for second level are (`\alph*`), for third level are `\roman*.` and for fourth level are `\Alph*.`.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

```
\NewDocumentCommand \itembx { 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={\itembx{#1}}` or `wrap-label={\itembx*{#1}}`.

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

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

3.2 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`.

3.2.1   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 values for first level 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`.

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 values for first level 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`.

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 are `2.0pt plus 1.0pt minus 1.0pt`, for third and fourth level are `1.0pt minus 1.0pt`.

💣 The value of this parameter also affects the *inner levels* and the `keyans` environment. 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 values for first level are `4.0pt plus 2.0pt minus 1.0pt`, for the rest of the levels are `2.0pt plus 1.0pt minus 1.0pt`.

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

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

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

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

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

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

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

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

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

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

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

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

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

3.2.2   Horizontal spaces

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

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

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

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

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

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

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

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

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

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

💣 If `list-indent=0pt` 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.

3.3 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 `\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*⟩}`\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`: `\list`{⟨*arg one*⟩}{⟨*arg two*⟩}{⟨*code*⟩}`\item`.

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

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

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

3.4 Keys for start, series and resume

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

Sets the *start value* of the numbering on the current level. Internally ⟨*string*⟩ is 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 *"first level"* of `enumext` and `enumext*` and are ignored if set when nested inside 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 the `start` key.

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 the `start` key.

💣 For security reasons the `series` key will never save in {⟨*series name*⟩} the keys `series`, `resume`, `resume*`, `save-ans`, `save-key` 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*.

3.5 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.

3.6 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`.

3.6.1 The command `\miniright`

`\miniright`
`\miniright*`

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 version* '*' 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.

3.6.2 The key `miniright`

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

`miniright` = {⟨*code for drawing or tabular*⟩}   default: *not used*

Set the *code* for the drawing or tabular to be placed in the `minipage` environment on the *"right side"* by starting it with `\centering`.

`miniright*` = {⟨*code for drawing or tabular*⟩}   default: *not used*

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

4 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 `keyans`, `keyans*` and `keyanspic` are available.

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

4.1 Keys for storage

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

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

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

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

Sets the *text symbol* that will separate the current ⟨*label*⟩ defined by the `label` key from the ⟨*optional argument*⟩ (if present), when storing them in the ⟨*store name*⟩ defined by the `save-ans` key for the `\item*` command in the `keyans` and `keyans*` environment and for the `\anspic` command in the `keyanspic` environment. 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.

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

Sets the *symbol* to be displayed in the left margin of the *"stored content"* in ⟨*store name*⟩ set by `save-ans` key when using `show-ans` key.

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.

4.2 Keys for internal `label` and `ref`

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

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

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

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

4.3 Keys for debugging and checking

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

Displays the *current* ⟨*argument*⟩ passed to `\anskey` in `enumext` environment, the current ⟨*label*⟩ for `\item*` in `keyans` environment and the current ⟨*label*⟩ for `\anspic*` in `keyanspic` environment at the place where it is executed. If the optional argument is present in `\item*` or `\anspic*` it will be shown in square brackets.

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

Displays the *position* occupied by the *"stored content"* by `\anskey` in `enumext` environment, `\item*` in `keyans` environment and `\anspic*` in `keyanspic` environment in ⟨*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. This key works under the logic that each question will contain *"only one answer"*, it is intended to be used in conjunction with `no-store` key.

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

This is a *meta-key* that does not receive an argument. This key is used in conjunction with `check-ans` and is designed to be used with nested levels of `enumext` in which the `\anskey` command will not be used.

4.4 The command `\anskey`

`\anskey`

`\anskey`{⟨*content*⟩}

The `\anskey` command takes a mandatory argument and is triggered by `save-ans` key. The *"content"* are *"stored"* in ⟨*store name*⟩ set by `save-ans` key. The command does *"not support"* verbatim content and must NOT be nested. By design it is assumed that each `\item` or `\item*` will have a *"single"* occurrence of the command unless a nested level is opened or the `no-store` key is used. If `save-ref` key are active and the `hyperref`[7] package is detected, `\hyperlink` and `\hypertarget` will be used, otherwise the usual *"label and ref"* system provided by LaTeX will be used.

Example

★ 1. Text containing our instructions or questions.   3. Text containing our instructions or questions.

   * ⎡first answer⎦   * ⎡third answer⎦

  2. Text containing our instructions or questions.   4. Text containing our instructions or questions.

   (a) Question.   * ⎡fourth answer⎦

      * ⎡second answer⎦

```
\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}
```

4.5 The environment keyans

keyans
keyans*

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

The keyans is an *"enumerated list"* environment designed for *"multiple choice"* questions activated by the save-ans key. This environment 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.

```
\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 environment and have higher precedence than those set by \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*).

4.5.1 The \item* in keyans

\item*

```
\item*
\item*[⟨content⟩]
```

The \item* and \item*[⟨content⟩] command store the current ⟨label⟩ set by label key next to the ⟨content⟩ (if it is present) in ⟨store name⟩ set by save-ans key in the *"first level"* of the enumext environment.

The *starred version* '*' 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 *starred version* '*' will only appear *"once"* within the environment.

💣 The behavior of \item* in keyans environment is NOT the same as in the enumext environment.

Example

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

2. Text containing a question and image.

   (A) Choice
   (B) Choice
   (C) Choice
   (D) Choice
   * (E) [note] Correct choice

Some text

4.6 The environment `keyanspic`

`keyanspic` \begin{keyanspic}[⟨*number above, number below*⟩]\anspic{⟨*drawing*⟩}\anspic*[⟨*content*⟩]{⟨*drawing*⟩}

The `keyanspic` is a *"fake enumerated list"* environment that which uses the `\anspic` command instead of `\item`. It is activated by the `save-ans` key and has the same settings as the `keyans` environment. It is intended for placing *"drawings"* or *"tabular"* with an in-line or *above* and *below* 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.

The optional argument determines the number drawings or tabular *"above"* and *"below"* within the environment. The vertical separation between *"above"* and *"below"* is controlled by the values set by `parsep` and `itemsep` keys passed to `keyans` environment. If the optional argument or the second part of it is omitted the drawings or tabular will be put on a single line.

4.6.1 The command `\anspic`

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

The `\anspic` command take three arguments, the *starred version* '`*`' store the current ⟨*label*⟩ next to the ⟨*content*⟩ (if it is present) in ⟨*store name*⟩ set by `save-ans` key.

The *starred version* '`*`' 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 version* '`*`' will only appear *"once"* within the environment.

Example

```
\begin{enumext}[save-ans=test,show-ans,nosep]
  \item Question with images.
    \begin{keyanspic}[3,2]
      \anspic{\includegraphics[scale=0.15]{example-image-a}}
      \anspic{\includegraphics[scale=0.15]{example-image-b}}
      \anspic{\includegraphics[scale=0.15]{example-image-a}}
      \anspic{\includegraphics[scale=0.15]{example-image-a}}
      \anspic*[note]{\includegraphics[scale=0.15]{example-image-a}}
    \end{keyanspic}
\end{enumext}
```
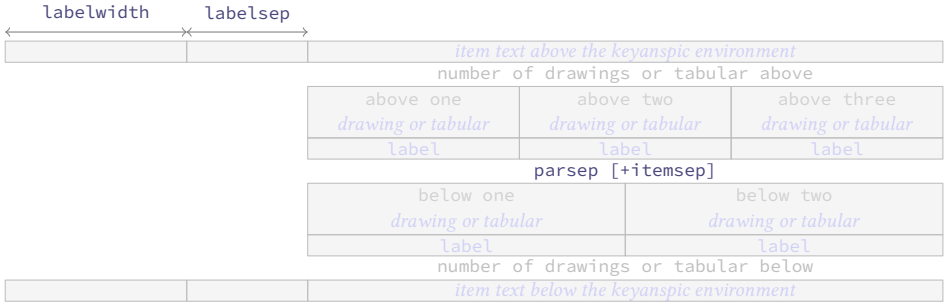
1. Question with images.

(A)

(B)

(C)

(D)

* (E)[note]
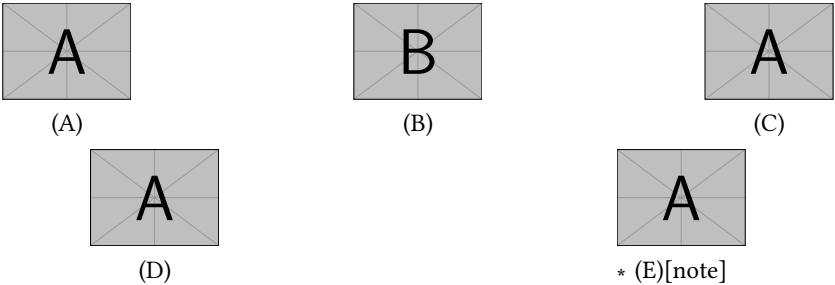
4.7 Printing stored content

4.7.1 The command \getkeyans

\getkeyans

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

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

The *"content"* can only be accessed *"after"* it is stored, if the ⟨*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.

4.7.2 The command \printkeyans

\printkeyans

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

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

Internally it places the *"stored content"* inside the enumext environment with default values for label key are the same as those of the enumext environment along with the keys: nosep, first=\small, font=\small for all levels, except for the first one that adds the columns=2 key.

The optional argument allows to handle the ⟨*keys*⟩ *"on the first level"* of the enumext environment encapsulated by the command. If need to pass options for nested levels use \setenumext[⟨*print , level*⟩]{⟨*store name*⟩}.

Example

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

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

  \item Related to Linux

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

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

\printkeyans{sample}
```

1. Factor $3x + 3y + 3z$.
   [1] $3(x + y + z)$
2. True False
   (a) LATEX2e is cool?
      [2] Very True!
3. Related to Linux
   (a) You use linux?

[3] Yes
(b) Rate the following package and class
   i. xsim
      [4] very good
   ii. exsheets
      [5] obsolete

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

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

5 Full examples

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

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

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

Example 1

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

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

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

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

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

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

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

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

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

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

Example 2

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

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

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

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

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

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

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

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

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

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

---

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

Example 3

A *"simple multiple choice"* test 📄.

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

Test keys

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

Example 4

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

🦆 1. Factor $x^2 - 2x + 1$

🦆 2. Factor $3x + 3y + 3z$

The following questions need to be cuaqtified :)

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

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

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

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

Example 5

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

1

Which choice best describes what happens in the passage?

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

3

Which choice best describes what happens in the passage?

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

2

Which choice best describes what happens in the passage?

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

4

Which choice best describes what happens in the passage?

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

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

6   The way of non-enumerated lists

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

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

Fake `itemize` environment

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

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

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

Fake `description` environment

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

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

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

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

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

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

Description indented by label

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

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

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

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

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

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

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

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

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

Description with multi-line labels

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

```
\NewDocumentCommand \itembx { 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*={\itembx{#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[4] 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`[10] and `l3seq`[10] modules together with the `hyperref`[7] and `enumitem`[5] 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.

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

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

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

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

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

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

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

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

[12] GUNDLACH, PATRICK. "The lua-visual-debug package". Available from CTAN, https://www.ctan.org/pkg/lua-visual-debug, 2023.

[13] LEMVIG, MOGENS. "The shortlst package". Available from CTAN, https://www.ctan.org/pkg/shortlst, 1998.

[14] NIEDERBERGER, CLEMENS. "tasks – Horizontally columned lists". Available from CTAN, https://www.ctan.org/pkg/tasks, 2022.

8 Change history

v1.0 2024-05-22   – First public release.

9 Index of Documentation

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

10 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.

10.1 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.

10.2 Initial set up

Start the DocStrip guards.

```
1 ⟨*package⟩
```

Identify the internal prefix (LaTeX3 DocStrip convention) for l3doc class.

```
2 ⟨@@=enumext⟩
```

10.3 Declaration of the package

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

```
3 \NeedsTeXFormat{LaTeX2e}[2023-11-01]
```

Now declare the enumext package.

```
4 \ProvidesExplPackage
5   {enumext}
6   {2024-05-22}
7   {1.0}
8   {Enumerate exercise sheets}
```

Finally check if the multicol package is 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}[2023-03-30]
18       }
19   }
```

10.4 Definition of variables

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

Integer variables will control the nesting levels of the environments and boolean variables will be used to determine if they are present (nested) in each other. The boolean variables \g__enumext_starred_bool and \g__enumext_standar_bool will be set to *"true"* when the enumext and enumext* environments are not nested with each other.

<div>

\l__enumext_level_int
\l__enumext_level_h_int
\l__enumext_keyans_level_int
\l__enumext_keyans_level_h_int
\l__enumext_keyans_pic_level_int
\l__enumext_starred_bool
\g__enumext_starred_bool
\l__enumext_starred_first_level_bool
\l__enumext_standar_bool
\g__enumext_standar_bool
\l__enumext_standar_first_level_bool
\l__enumext_keyans_env_bool

</div>

```
20 \int_new:N  \l__enumext_level_int
21 \int_new:N  \l__enumext_level_h_int
22 \int_new:N  \l__enumext_keyans_level_int
23 \int_new:N  \l__enumext_keyans_level_h_int
24 \int_new:N  \l__enumext_keyans_pic_level_int
25 \bool_new:N \l__enumext_starred_bool
26 \bool_new:N \g__enumext_starred_bool
```

```
27 \bool_new:N \l__enumext_starred_first_level_bool
28 \bool_new:N \l__enumext_standar_bool
29 \bool_new:N \g__enumext_standar_bool
30 \bool_new:N \l__enumext_standar_first_level_bool
31 \bool_new:N \l__enumext_keyans_env_bool
```

(*End of definition for* \l__enumext_level_int *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 (§10.8) and then modified by the function __enumext_label_style:Nnn used by label key (§10.11).

```
32 \cs_set_protected:Npn \__enumext_tmp:n #1
33   {
34     \tl_new:c { l__enumext_counter_#1_tl }
35   }
36 \clist_map_inline:nn { i, ii, iii, iv, v, vi, vii, viii } { \__enumext_tmp:n {#1} }
```

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

\c__enumext_counter_style_tl
\l__enumext_ref_key_arg_tl
\l__enumext_ref_the_count_tl
\l__enumext_the_counter_X_tl
\l__enumext_renew_the_count_X_tl

Internal variables used by ref key (§10.11).

```
37 \tl_const:Nn \c__enumext_counter_style_tl
38   { { arabic } { roman } { Roman } { alph } { Alph } }
39 \tl_new:N \l__enumext_ref_key_arg_tl
40 \tl_new:N \l__enumext_ref_the_count_tl
41 \cs_set_protected:Npn \__enumext_tmp:n #1
42   {
43     \tl_new:c  { l__enumext_renew_the_count_#1_tl }
44     \tl_new:c  { l__enumext_the_counter_#1_tl }
45     \tl_set:ce { l__enumext_the_counter_#1_tl } { \exp_not:c { theenumX#1 } }
46   }
47 \clist_map_inline:nn { i, ii, iii, iv, v, vi, vii, viii } { \__enumext_tmp:n {#1} }
```

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

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

Internal variables used by resume, resume* and series keys. The global token list \g__enumext_-item_symbol_tl is used by item-sym* key (§10.27).

```
48 \int_new:N  \g__enumext_resume_int
49 \int_new:N  \g__enumext_resume_vii_int
50 \tl_new:N   \l__enumext_resume_name_tl
51 \bool_new:N \l__enumext_resume_active_bool
52 \tl_new:N   \g__enumext_item_symbol_tl
53 \tl_new:N   \g__enumext_standar_series_tl
54 \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 (§10.12) and label (§10.10) keys.

```
55 \dim_new:N \l__enumext_current_widest_dim
56 \tl_new:N  \g__enumext_counter_styles_tl
57 \tl_new:N  \g__enumext_widest_label_tl
58 \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 (§10.14).

The variables \l__enumext_leftmargin_X_dim and \l__enumext_itemindent_X_dim are used (and set) by the function __enumext_calc_hspace:NNNNNNNNNNNN (§10.31.1) which determines the internal values for \leftmargin and \itemindent.

```
59 \cs_set_protected:Npn \__enumext_tmp:n #1
60   {
61     \bool_new:c { l__enumext_leftmargin_tmp_#1_bool }
62     \dim_new:c  { l__enumext_leftmargin_tmp_#1_dim }
63     \dim_new:c  { l__enumext_leftmargin_#1_dim       }
64     \dim_new:c  { l__enumext_itemindent_#1_dim       }
65   }
66 \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.)*

Internal variables used by columns key §10.18).

```
67  \cs_set_protected:Npn \__enumext_tmp:n #1
68    {
69      \skip_new:c  { l__enumext_multicols_above_#1_skip }
70      \skip_new:c  { l__enumext_multicols_below_#1_skip }
71    }
72  \clist_map_inline:nn { i, ii, iii, iv, v } { \__enumext_tmp:n {#1} }
```

\l__enumext_multicols_above_X_skip
\l__enumext_multicols_below_X_skip

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

Internal variables used by `\miniright` command (§10.19.4) and the keys miniright, miniright*, mini-env and mini-sep (§10.17, §10.19).

```
73  \int_new:N  \g__enumext_minipage_stat_int
74  \skip_new:N \l__enumext_minipage_left_skip
75  \skip_new:N \l__enumext_minipage_right_skip
76  \skip_new:N \l__enumext_minipage_after_skip
77  \skip_new:N \g__enumext_minipage_right_skip
78  \skip_new:N \g__enumext_minipage_after_skip
79  \cs_set_protected:Npn \__enumext_tmp:n #1
80    {
81      \dim_new:c  { l__enumext_minipage_left_#1_dim    }
82      \bool_new:c { l__enumext_minipage_active_#1_bool }
83    }
84  \clist_map_inline:nn { i, ii, iii, iv, v, vii, viii } { \__enumext_tmp:n {#1} }
```

\g__enumext_minipage_stat_int
\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

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

The integer variable `\l__enumext_start_X_int` are used by the start key (§10.12), the token list `\l__enumext_fake_item_indent_X_tl` is used by itemindent key, the variables `\l__enumext_label_fill_left_X_tl` and `\l__enumext_label_fill_left_X_tl` are used by the align key (§10.10). 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

```
85  \cs_set_protected:Npn \__enumext_tmp:n #1
86    {
87      \bool_new:c { l__enumext_wrap_label_#1_bool     }
88      \bool_new:c { l__enumext_wrap_label_opt_#1_bool }
89      \int_new:c  { l__enumext_start_#1_int           }
90      \tl_new:c   { l__enumext_fake_item_indent_#1_tl }
91      \tl_new:c   { l__enumext_label_fill_left_#1_tl  }
92      \tl_new:c   { l__enumext_label_fill_right_#1_tl }
93      \bool_new:c { l__enumext_vspace_a_star_#1_bool  }
94      \bool_new:c { l__enumext_vspace_b_star_#1_bool  }
95    }
96  \clist_map_inline:nn { i, ii, iii, iv, v, vii, viii } { \__enumext_tmp:n {#1} }
```

\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

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

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

The variable `\l__enumext_store_name_tl` sets the name for the storage in ⟨*sequence*⟩ and ⟨*prop list*⟩, the variable `\g__enumext_store_name_tl` is just a copy of the storage name used by the check-ans key (§??).

The variable `\l__enumext_store_anskey_arg_tl` stores the contents of `\anskey` (§10.25) and the variable `\l__enumext_store_keyans_label_tl` stores the contents of `\item*` (§10.29.2) for the keyans and keyans* environments and the contents of `\anspic*` (§10.34.1) for the keyanspic environment.

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

```
 97  \bool_new:N \l__enumext_store_active_bool
 98  \tl_new:N   \l__enumext_store_name_tl
 99  \tl_new:N   \g__enumext_store_name_tl
100  \tl_new:N   \l__enumext_store_anskey_arg_tl
101  \int_new:N  \l__enumext_store_columns_join_int
102  \tl_new:N   \l__enumext_store_keyans_label_tl
103  \tl_new:N   \l__enumext_store_keyans_item_opt_tl
104  \tl_new:N   \l__enumext_keyans_item_opt_tl
105  \tl_new:N   \l__enumext_keyans_tmpa_tl
```

\l__enumext_store_active_bool
\l__enumext_store_name_tl
\g__enumext_store_name_tl
\l__enumext_store_anskey_arg_tl
\l__enumext_store_columns_join_int
\l__enumext_store_keyans_label_tl
\l__enumext_store_keyans_item_opt_tl
\l__enumext_keyans_item_opt_tl
\l__enumext_keyans_tmpa_tl

(End of definition for \l__enumext_store_active_bool and others.)

\l__enumext_setkey_tmpa_tl
\l__enumext_setkey_tmpb_tl
\l__enumext_setkey_tmpa_int
\l__enumext_setkey_tmpa_seq
\l__enumext_setkey_tmpb_seq

Internal variables used by the command \setenumext (§10.39).

```
106 \tl_new:N  \l__enumext_setkey_tmpa_tl
107 \tl_new:N  \l__enumext_setkey_tmpb_tl
108 \int_new:N \l__enumext_setkey_tmpa_int
109 \seq_new:N \l__enumext_setkey_tmpa_seq
110 \seq_new:N \l__enumext_setkey_tmpb_seq
```

(End of definition for \l__enumext_setkey_tmpa_tl and others.)

\l__enumext_store_opt_X_tl
\l__enumext_print_keyans_X_tl
\l__enumext_store_columns_X_bool
\l__enumext_store_columns_X_int
\l__enumext_store_columns_sep_X_bool
\l__enumext_store_columns_sep_X_dim
\l__enumext_store_upper_level_X_bool

Internal variables used by [⟨*key = val*⟩] in enumext and enumext* environment, the command \printkeyans (§10.38) and the keys columns* and columns-sep*.

```
111 \cs_set_protected:Npn \__enumext_tmp:n #1
112   {
113     \tl_new:c { l__enumext_store_opt_#1_tl              }
114     \tl_new:c { l__enumext_print_keyans_#1_tl           }
115     \bool_new:c { l__enumext_store_columns_#1_bool      }
116     \int_new:c { l__enumext_store_columns_#1_int        }
117     \bool_new:c { l__enumext_store_columns_sep_#1_bool }
118     \dim_new:c { l__enumext_store_columns_sep_#1_dim    }
119     \bool_new:c { l__enumext_store_upper_level_#1_bool }
120   }
121 \clist_map_inline:nn { i, ii, iii, iv, vii } { \__enumext_tmp:n {#1} }
```

(End of definition for \l__enumext_store_opt_X_tl and others.)

\l__enumext_show_answer_bool
\l__enumext_show_position_bool
\l__enumext_mark_ref_sym_tl
\l__enumext_mark_answer_sym_tl
\l__enumext_mark_position_str

Internal variables for *"storage system"* mechanism used by \anskey (§10.25), keyans and keyanspic environments. These variables are used by show-ans, show-pos, mark-ans, save-key and mark-ref keys (§10.24).

```
122 \bool_new:N \l__enumext_show_answer_bool
123 \bool_new:N \l__enumext_show_position_bool
124 \tl_new:N   \l__enumext_mark_ref_sym_tl
125 \tl_new:N   \l__enumext_mark_answer_sym_tl
126 \str_new:N  \l__enumext_mark_position_str
```

(End of definition for \l__enumext_show_answer_bool and others.)

\l__enumext_keyans_pic_body_seq
\l__enumext_keyans_pic_width_dim
\l__enumext_keyans_pic_above_int
\l__enumext_keyans_pic_below_int
\l__enumext_keyans_pic_above_skip

Internal variables used by keyanspic environment (§10.34.2).

```
127 \seq_new:N  \l__enumext_keyans_pic_body_seq
128 \dim_new:N  \l__enumext_keyans_pic_width_dim
129 \int_new:N  \l__enumext_keyans_pic_above_int
130 \int_new:N  \l__enumext_keyans_pic_below_int
131 \skip_new:N \l__enumext_keyans_pic_above_skip
```

(End of definition for \l__enumext_keyans_pic_body_seq and others.)

\l__enumext_store_ans_bool
\l__enumext_check_ans_bool
\g__enumext_check_ans_bool
\l__enumext_check_start_line_env_tl
\g__enumext_check_start_line_env_tl
\g__enumext_check_starred_cmd_int
\g__enumext_count_item_anskey_int
\g__enumext_count_item_number_int

Internal variables used by *"check answer"* mechanism (§10.23) 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.

```
132 \bool_new:N \l__enumext_store_ans_bool
133 \bool_new:N \l__enumext_check_ans_bool
134 \bool_new:N \g__enumext_check_ans_bool
135 \tl_new:N   \l__enumext_check_start_line_env_tl
136 \tl_new:N   \g__enumext_check_start_line_env_tl
137 \tl_new:N   \g__enumext_check_environment_name_tl
138 \int_new:N  \g__enumext_check_starred_cmd_int
139 \int_new:N  \g__enumext_count_item_anskey_int
140 \int_new:N  \g__enumext_count_item_number_int
```

(End of definition for \l__enumext_store_ans_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 (§10.7). The boolean variable \l__enumext_footnotes_key_bool determine if hyperref is load with key hyperfootnotes=true.

```
141 \bool_new:N \l__enumext_hyperref_bool
142 \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_store_write_aux_file_tl
\l__enumext_label_copy_X_tl

Internal variables are used when executing the `save-ref` key. 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` and the variable `\l__-enumext_store_write_aux_file_tl` will be in charge of executing the writing code in the `.aux` file.

```
143 \tl_new:N \l__enumext_newlabel_arg_one_tl
144 \tl_new:N \l__enumext_newlabel_arg_two_tl
145 \tl_new:N \l__enumext_store_write_aux_file_tl
146 \cs_set_protected:Npn \__enumext_tmp:n #1
147   {
148     \tl_new:c { l__enumext_label_copy_#1_tl }
149   }
150 \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`.

```
151 \int_new:N \g__enumext_footnote_int
152 \seq_new:N \g__enumext_footnote_arg_seq
153 \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_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
\g__enumext_miniright_code_X_tl
\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.

```
154 \cs_set_protected:Npn \__enumext_tmp:n #1
155   {
156     \bool_new:c { l__enumext_item_starred_#1_bool    }
157     \int_new:c  { l__enumext_item_column_pos_#1_int  }
158     \int_new:c  { g__enumext_item_count_all_#1_int   }
159     \int_new:c  { l__enumext_joined_item_#1_int      }
160     \int_new:c  { l__enumext_joined_item_aux_#1_int  }
161     \int_new:c  { l__enumext_tmpa_#1_int             }
162     \box_new:c  { l__enumext_item_text_#1_box        }
163     \dim_new:c  { l__enumext_joined_width_#1_dim     }
164     \dim_new:c  { l__enumext_item_width_#1_dim       }
165     \tl_new:c   { g__enumext_item_symbol_aux_#1_tl   }
166     \str_new:c  { l__enumext_align_label_#1_str      }
167     \bool_new:c { g__enumext_minipage_active_#1_bool }
168     \tl_new:c   { g__enumext_miniright_code_#1_tl    }
169     \bool_new:c { g__enumext_minipage_center_#1_bool }
170     \dim_new:c  { g__enumext_minipage_right_#1_dim   }
171     \skip_new:c { g__enumext_minipage_right_#1_skip  }
172   }
173 \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`.

```
174 \clist_const:Nn \c__enumext_all_envs_clist
175   {
176     {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv},
177     {keyans}{v}, {enumext*}{vii}, {keyans*}{viii}
178   }
```

(*End of definition for* \c__enumext_all_envs_clist.)

10.5   Some utility functions

__enumext_at_begin_document:n

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

```
179 \cs_new_protected:Npn \__enumext_at_begin_document:n #1
180   {
181     \hook_gput_code:nnn {begindocument} {enumext} { #1 }
182   }
```

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

`\__enumext_after_env:nn`

A internal *"hook"* function for execute code `minirigth` and `minirigth*` keys outside the `enumext*` and `keyans*` environments and print `check-ans` outside the `enumext` and `enumext*` environments.

```
183 \cs_new_protected:Npn \__enumext_after_env:nn #1 #2
184   {
185     \hook_gput_code:nnn {env/#1/after} {enumext} {#2}
186   }
```

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

`\__enumext_level:`

Function for check current level in `enumext`.

```
187 \cs_new:Nn \__enumext_level:
188   {
189     \int_to_roman:n { \l__enumext_level_int }
190   }
```

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

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

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

```
191 \prg_new_protected_conditional:Npnn \__enumext_if_is_int:n #1 { T, F, TF }
192   {
193     \regex_match:nnTF { ^[\+\-]?[\d]+$ } {#1} % $
194       { \prg_return_true: }
195       { \prg_return_false: }
196   }
```

(*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.

```
197 \cs_new_protected:Nn \__enumext_regex_counter_style:
198   {
199     \tl_map_inline:Nn \c__enumext_counter_style_tl
200       {
201         \regex_replace_once:nnN { \c{##1}\* }
202           { \c{##1}\cB{\u{l__enumext_ref_the_count_tl}\cE} } \l__enumext_ref_key_arg_tl
203       }
204   }
```

(*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.

```
205 \cs_new:Npn \__enumext_show_length:nnn #1 #2 #3
206   {
207     * ~ #2
208     \prg_replicate:nn { 14 - \str_count:n {#2} } { ~ }
209       = ~ \use:c { #1_use:c } { l__enumext_#2_#3_#1 } \\
210   }
```

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

`\__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.

```
211 \cs_new_protected:Nn \__enumext_is_not_nested:
212   {
213     \str_case:en { \@currenvir }
214       {
215         {enumext}
216           {
217             \bool_lazy_and:nnT
218               { \bool_not_p:n { \g__enumext_standar_bool } }
219               { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
220               {
221                 \bool_gset_true:N \g__enumext_standar_bool
222               }
```

```
223                }
224            {enumext*}
225                {
226                    \bool_lazy_and:nnT
227                        { \bool_not_p:n { \g__enumext_starred_bool } }
228                        { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
229                        {
230                            \bool_gset_true:N \g__enumext_starred_bool
231                        }
232                }
233            }
234        }
```

The function __enumext_is_on_first_level: will set the variables \l__enumext_standard_-
first_level_bool and \l__enumext_standard_first_level_bool to "*true*" only if the environ-
ment 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_check_start_line_env_tl to use in messages related to
the check-ans key.

```
235  \cs_new_protected:Nn \__enumext_is_on_first_level:
236    {
237      \bool_lazy_all:nT
238        {
239          { \bool_if_p:N \g__enumext_standar_bool }
240          { \int_compare_p:nNn { \l__enumext_level_int } = { 1 } }
241          { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
242        }
243        {
244          \bool_set_true:N \l__enumext_standar_first_level_bool
245          \tl_gset:Nn \g__enumext_check_environment_name_tl { enumext }
246          \tl_gset:Ne \g__enumext_check_start_line_env_tl
247            {
248              in ~ '\exp_not:V \g__enumext_check_environment_name_tl' ~
249              start ~ on ~ line ~ \exp_not:V \inputlineno
250            }
251        }
252      \bool_lazy_all:nT
253        {
254          { \bool_if_p:N \g__enumext_starred_bool }
255          { \int_compare_p:nNn { \l__enumext_level_h_int } = { 1 } }
256          { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
257        }
258        {
259          \bool_set_true:N \l__enumext_starred_first_level_bool
260          \tl_gset:Nn \g__enumext_check_environment_name_tl { enumext* }
261          \tl_gset:Ne \g__enumext_check_start_line_env_tl
262            {
263              in ~ '\exp_not:V \g__enumext_check_environment_name_tl' ~
264              start ~ on ~ line ~ \exp_not:V \inputlineno
265            }
266        }
267    }
```

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

__enumext_keyans_save_start_line: The function __enumext_keyans_save_start_line: will save the start line number of the environ-
ments keyans, keyans* and keyanspic in the variable \l__enumext _check_start_line_env_tl
to use in the __enumext_check_starred_cmd:n function.

```
268  \cs_new_protected:Nn \__enumext_keyans_save_start_line:
269    {
270      \str_case:en { \@currenvir }
271        {
272          {keyans}
273            {
274              \tl_set:Ne \l__enumext_check_start_line_env_tl
275                {
276                  in ~ 'keyans' ~ start ~ on ~ line ~ \exp_not:V \inputlineno
277                }
278            }
279          {keyans*}
280            {
281              \tl_set:Ne \l__enumext_check_start_line_env_tl
```

```
282             {
283               in ~ 'keyans*' ~ start ~ on ~ line ~ \exp_not:V \inputlineno
284             }
285           }
286         {keyanspic}
287           {
288             \tl_set:Ne \l__enumext_check_start_line_env_tl
289               {
290                 in ~ 'keyanspic' ~ start ~ on ~ line ~ \exp_not:V \inputlineno
291               }
292           }
293       }
294   }
```

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

`\__enumext_execute_after_env:` The function `\__enumext_execute_after_env:` will perform the comparison between the `\item` in the environments and the `\item`'s with answers and return the appropriate message. As this function is passed to the function `\__enumext_after_env:nn` for the environments `enumext` and `enumext*` we must make sure that we are not nested at any level and finally reset our global variables.

```
295 \cs_new_protected:Nn \__enumext_execute_after_env:
296   {
297     \int_compare:nNnT { \l__enumext_level_int } = { 0 }
298       {
299         \tl_if_empty:NF \g__enumext_store_name_tl
300           {
301
302           }
303         \int_gzero:N \g__enumext_count_item_number_int
304         \int_gzero:N \g__enumext_count_item_anskey_int
305         \bool_gset_false:N \g__enumext_check_ans_bool
306         \bool_gset_false:N \g__enumext_standar_bool
307         \bool_gset_false:N \g__enumext_starred_bool
308         \tl_gclear:N \g__enumext_store_name_tl
309         \tl_gclear:N \g__enumext_check_start_line_env_tl
310         \tl_gclear:N \g__enumext_check_environment_name_tl
311       }
312   }
```

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

10.6   Copying `list` and `minipage` environments

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

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

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

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

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

```
313 \__enumext_at_begin_document:n
314   {
315     \cs_new_eq:NN \__enumext_start_list:nn \list
316     \cs_new_eq:NN \__enumext_stop_list: \endlist
317     \cs_new_eq:NN \__enumext_item_std:w \item
318   }
```

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

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

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

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

`\__enumext_minipage:w`
`\__enumext_endminipage:`

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

```
319 \__enumext_at_begin_document:n
320   {
321     \cs_new_eq:NN \__enumext_minipage:w \minipage
322     \cs_new_eq:NN \__enumext_endminipage: \endminipage
323   }
```

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

10.7 Compatibility with hyperref and footnotehyper

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

```
324 \hook_gput_code:nnn { begindocument } { enumext } { \__enumext_after_hyperref: }
325 \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".

```
326 \cs_new_protected:Nn \__enumext_after_hyperref:
327   {
328     \IfPackageLoadedTF { hyperref }
329       {
330         \msg_info:nnn { enumext } { package-load } { hyperref }
331         \bool_set_true:N \l__enumext_hyperref_bool
332         \IfHyperBoolean{hyperfootnotes}
333           {
334             \typeout{hyperfootnotes=true}
335             \bool_set_true:N \l__enumext_footnotes_key_bool
336           }
337           { \typeout{hyperfootnotes=false} }
338       }
339       {  }
```

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`.

```
340     \bool_if:NT \l__enumext_footnotes_key_bool
341       {
342         \IfPackageLoadedTF { footnotehyper }
343           {
344             \msg_info:nnn { enumext } { package-load } { footnotehyper }
345           }
346           {
347             \typeout{No ~ footnotehyper ~ load}
348             \typeout{Load ~ and  ~ use  ~ \string\makesavenoteenv{enumext*}}
349             \bool_set_false:N \l__enumext_footnotes_key_bool
350           }
351       }
```

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.

```
352     \bool_if:NTF \l__enumext_hyperref_bool
353       {
354         \cs_new_eq:NN \__enumext_hypertarget:nn \hypertarget
355         \cs_new_eq:NN \__enumext_phantomsection: \phantomsection
356       }
357       {
358         \cs_new_eq:NN \__enumext_hypertarget:nn \use_none:nn
359         \cs_new_eq:NN \__enumext_phantomsection: \prg_do_nothing:
360       }
361   }
```

(*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.

```
362 \cs_new_protected:Npn \__enumext_newlabel:nn #1 #2
363   {
364     \protected@write \@auxout { }
365       {
366         \token_to_str:N \newlabel {#1}
367           {
368             {#2}
369             \bool_if:NT \l__enumext_hyperref_bool
370               { { \thepage } {#2} {#1} }
371             { }
372           }
373       }
374     \__enumext_hypertarget:nn {#1} { }
375     \__enumext_phantomsection:
376   }
```

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

10.8   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.

```
377 \cs_new_protected:Npn \__enumext_define_counters:Nn #1 #2
378   {
379     \cs_if_exist:cTF { c@ #2 }
380       { \msg_fatal:nnn { enumext } { counters }{ #2 } }
381       {
382         \tl_set:Nn #1 { #2 }
383         \newcounter { #2 }
384       }
385   }
```

(*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.

```
386 \__enumext_define_counters:Nn \l__enumext_counter_i_tl    { enumXi     }
387 \__enumext_define_counters:Nn \l__enumext_counter_ii_tl   { enumXii    }
388 \__enumext_define_counters:Nn \l__enumext_counter_iii_tl  { enumXiii   }
389 \__enumext_define_counters:Nn \l__enumext_counter_iv_tl   { enumXiv    }
390 \__enumext_define_counters:Nn \l__enumext_counter_v_tl    { enumXv     }
391 \__enumext_define_counters:Nn \l__enumext_counter_vi_tl   { enumXvi    }
392 \__enumext_define_counters:Nn \l__enumext_counter_vii_tl  { enumXvii   }
393 \__enumext_define_counters:Nn \l__enumext_counter_viii_tl { enumXviii  }
```

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

10.9   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 environment and the keyans environment, so it is necessary to get a default value for labelwidth from these ⟨*labels*⟩ at the same time.

```
394 \cs_new_protected:Npn \__enumext_register_counter_style:Nn #1 #2
395   {
396     \tl_const:cn { c__enumext_widest_ \cs_to_str:N #1 _tl } {#2}
397     \tl_gput_right:Nn \g__enumext_counter_styles_tl {#1}
398   }
399 \__enumext_register_counter_style:Nn \arabic { 0 }
400 \__enumext_register_counter_style:Nn \Alph  { M }
401 \__enumext_register_counter_style:Nn \alph  { m }
402 \__enumext_register_counter_style:Nn \Roman { VIII }
403 \__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.

```
404  \cs_new_protected:Npn \__enumext_label_width_by_box:Nn #1 #2
405    {
406      \hbox_set:Nn \l__enumext_label_width_by_box {#2}
407      \dim_set:Nn #1 { \box_wd:N \l__enumext_label_width_by_box }
408    }
409  \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.

```
410  \cs_new_protected:Npn \__enumext_label_style:Nnn #1 #2 #3
411    {
412      \tl_clear_new:N #1
413      \tl_put_right:Ne #1 { \tl_trim_spaces:n {#3} }
414      \tl_gset_eq:NN \g__enumext_widest_label_tl #1
415      \tl_map_inline:Nn \g__enumext_counter_styles_tl
416        {
417          \tl_replace_all:Nne #1 { ##1* } { \exp_not:N ##1 {#2} }
418          \tl_greplace_all:Nne \g__enumext_widest_label_tl { ##1* }
419            { \tl_use:c { c__enumext_widest_ \cs_to_str:N ##1 _tl } }
420        }
421      \__enumext_label_width_by_box:Nn \l__enumext_current_widest_dim
422        { \tl_use:N \g__enumext_widest_label_tl }
423      \tl_set_eq:cN { the #2 } #1
424    }
425  \cs_generate_variant:Nn \__enumext_label_style:Nnn { cvn }
```

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

10.10 Setting keys associated with label

font
labelsep
labelwidth
wrap-label
wrap-label*

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

```
426  \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
427    {
428      \keys_define:nn { enumext / #1 }
429        {
430          font       .tl_set:c   = { l__enumext_label_font_style_#2_tl },
431          font       .value_required:n = true,
432          labelsep   .dim_set:c  = { l__enumext_labelsep_#2_dim },
433          labelsep   .initial:n  = {0.3333em},
434          labelsep   .value_required:n = true,
435          labelwidth .dim_set:c  = { l__enumext_labelwidth_#2_dim },
436          labelwidth .value_required:n = true,
437          wrap-label  .cs_set_protected:cp = { __enumext_wrapper_label_#2:n } ##1,
438          wrap-label  .initial:n  = {##1},
439          wrap-label  .value_required:n = true,
440          wrap-label* .code:n = {
441                          \bool_set_true:c { l__enumext_wrap_label_opt_#2_bool }
442                          \keys_set:nn { enumext / #1 } { wrap-label = {##1} }
443                        },
444          wrap-label* .value_required:n = true,
445        }
446    }
447  \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.

```
448  \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
449    {
450      \keys_define:nn { enumext / #1 }
451        {
452          align .choice:,
453          align / left   .code:n =
454                            {
455                              \tl_clear:c { l__enumext_label_fill_left_#2_tl  }
456                              \tl_set:cn  { l__enumext_label_fill_right_#2_tl } { \hfill }
457                            },
458          align / right  .code:n =
459                            {
460                              \tl_set:cn  { l__enumext_label_fill_left_#2_tl  } { \hfill }
461                              \tl_clear:c { l__enumext_label_fill_right_#2_tl }
462                            },
463          align / center .code:n =
464                            {
465                              \tl_set:cn { l__enumext_label_fill_left_#2_tl  } { \hfill }
466                              \tl_set:cn { l__enumext_label_fill_right_#2_tl } { \hfill }
467                            },
468          align .initial:n  = left,
469          align .value_required:n  = true,
470        }
471    }
472  \clist_map_inline:nn
473    {
474      {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {keyans}{v}
475    }
476    { \__enumext_tmp:nn #1 }

477  \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
478    {
479      \keys_define:nn { enumext / #1 }
480        {
481          align .choice:,
482          align / left   .code:n = \str_set:cn { l__enumext_align_label_#2_str } { l },
483          align / right  .code:n = \str_set:cn { l__enumext_align_label_#2_str } { r },
484          align / center .code:n = \str_set:cn { l__enumext_align_label_#2_str } { c },
485          align .initial:n = left,
486          align .value_required:n = true,
487        }
488    }
489  \clist_map_inline:nn { {enumext*}{vii}, {keyans*}{viii} } { \__enumext_tmp:nn #1 }
```

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

10.11 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.

10.11.1 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.

```
490  \cs_set_protected:Npn \__enumext_tmp:nnn #1 #2 #3
491    {
492      \keys_define:nn { enumext / #1 }
493        {
494          label .code:n      = {
495                              \__enumext_label_style:cvn { l__enumext_label_#2_tl }
496                                { l__enumext_counter_#2_tl } {##1}
497                              \dim_set_eq:cN  { l__enumext_labelwidth_#2_dim }
498                                \l__enumext_current_widest_dim
499                            },
500          label .initial:n = #3,
501          label .value_required:n = true,
502          ref   .code:n      = \__enumext_standar_ref:n {##1},
503          ref   .value_required:n = true,
504        }
```

```
505    }
506  \__enumext_tmp:nnn { level-1 } {   i } { \arabic*.}
507  \__enumext_tmp:nnn { level-2 } {  ii } { (\alph*) }
508  \__enumext_tmp:nnn { level-3 } { iii } { \roman*. }
509  \__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.

```
510  \cs_new_protected:Npn \__enumext_standar_ref:n #1
511    {
512      \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
513      \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
514        {
515          \msg_error:nnn { enumext } { key-ref-empty } { enumext }
516        }
517        {
518          \tl_set_eq:Nc
519            \l__enumext_ref_the_count_tl { l__enumext_counter_ \__enumext_level: _tl }
520          \__enumext_regex_counter_style:
521          \tl_set_eq:Nc
522            \l__enumext_ref_the_count_tl { l__enumext_the_counter_ \__enumext_level: _tl }
523          \tl_put_right:ce { l__enumext_renew_the_count_ \__enumext_level: _tl }
524            {
525              \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl }
526                { \exp_not:V \l__enumext_ref_key_arg_tl }
527            }
528        }
529    }
```

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

```
530  \cs_new_protected:Nn \__enumext_standar_ref:
531    {
532      \tl_if_empty:cF { l__enumext_renew_the_count_ \__enumext_level: _tl }
533        {
534          \tl_use:c { l__enumext_renew_the_count_ \__enumext_level: _tl }
535        }
536    }
```

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

10.11.2   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.

```
537  \cs_set_protected:Npn \__enumext_tmp:nnn #1 #2 #3
538    {
539      \keys_define:nn { enumext / #1 }
540        {
541        label .code:n    = {
542                            \__enumext_label_style:cvn { l__enumext_label_#2_tl }
543                              { l__enumext_counter_#2_tl } {##1}
544                            \dim_set_eq:cN  { l__enumext_labelwidth_#2_dim }
545                              \l__enumext_current_widest_dim
546                           },
547        label .initial:n = #3,
548        label .value_required:n = true,
549        ref    .code:n    = \__enumext_starred_ref:n {##1},
550        ref    .value_required:n = true,
551        }
552    }
553  \__enumext_tmp:nnn { enumext* } {  vii } { \arabic*.}
554  \__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.

```
555  \cs_new_protected:Npn \__enumext_starred_ref:n #1
556    {
557      \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
558      \int_compare:nNnT { \l__enumext_level_h_int } = { 1 }
559        {
560          \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
561            {
562              \msg_error:nnn { enumext } { key-ref-empty } { enumext* }
563            }
564            {
565              \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_counter_vii_tl
566              \__enumext_regex_counter_style:
567              \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_the_counter_vii_tl
568              \tl_put_right:Ne \l__enumext_renew_the_count_vii_tl
569                {
570                  \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl }
571                    { \exp_not:V \l__enumext_ref_key_arg_tl }
572                }
573            }
574        }
575      \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
576        {
577          \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
578            {
579              \msg_error:nnn { enumext } { key-ref-empty } { keyans* }
580            }
581            {
582              \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_counter_viii_tl
583              \__enumext_regex_counter_style:
584              \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_the_counter_viii_tl
585              \tl_put_right:Ne \l__enumext_renew_the_count_viii_tl
586                {
587                  \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl }
588                    { \exp_not:V \l__enumext_ref_key_arg_tl }
589                }
590            }
591        }
592    }
```

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.

```
593  \cs_new_protected:Nn \__enumext_starred_ref:
594    {
595      \int_compare:nNnT { \l__enumext_level_h_int } = { 1 }
596        {
597          \tl_if_empty:NF \l__enumext_renew_the_count_vii_tl
598            {
599              \tl_use:N \l__enumext_renew_the_count_vii_tl
600            }
601        }
602      \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
603        {
604          \tl_if_empty:NF \l__enumext_renew_the_count_viii_tl
605            {
606              \tl_use:N \l__enumext_renew_the_count_viii_tl
607            }
608        }
609    }
```

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

10.11.3 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.

```
610  \keys_define:nn { enumext / keyans }
611    {
612      label .code:n    = {
613                          \__enumext_label_style:cvn { l__enumext_label_v_tl }
614                            { l__enumext_counter_v_tl } {#1}
```

```
615                              \dim_set_eq:cN  { l__enumext_labelwidth_v_dim }
616                                \l__enumext_current_widest_dim
617                              \__enumext_label_style:cvn { l__enumext_label_vi_tl }
618                                { l__enumext_counter_vi_tl } {#1}
619                              \dim_set_eq:cN  { l__enumext_labelwidth_v_dim }
620                                \l__enumext_current_widest_dim
621                            },
622      label .initial:n = (\Alph*),
623      label .value_required:n = true,
624      ref    .code:n    = \__enumext_keyans_ref:n {#1},
625      ref    .value_required:n = true,
626    }
```

(*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.

```
627 \cs_new_protected:Npn \__enumext_keyans_ref:n #1
628    {
629      \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
630      \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
631        {
632          \msg_error:nnn { enumext } { key-ref-empty } { keyans }
633        }
634        {
635          \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_counter_v_tl
636          \__enumext_regex_counter_style:
637          \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_the_counter_v_tl
638          \tl_put_right:Ne \l__enumext_renew_the_count_v_tl
639            {
640              \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl }
641                { \exp_not:V \l__enumext_ref_key_arg_tl }
642            }
643        }
644    }
```

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

```
645 \cs_new_protected:Nn \__enumext_keyans_ref:
646    {
647      \tl_if_empty:NF \l__enumext_renew_the_count_v_tl
648        {
649          \tl_use:N \l__enumext_renew_the_count_v_tl
650        }
651    }
```

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

10.12 Setting start and widest keys

__enumext_start_from:NNn
__enumext_start_from:ccn

The function __enumext_start_from:NNn used by the start key 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.

```
652 \cs_new_protected:Npn \__enumext_start_from:NNn #1 #2 #3
653    {
654      \__enumext_if_is_int:nTF { #3 }
655        {
656          \int_set:Nn #2 {#3}
657        }
658        {
659          \regex_match:nVT { \c{Alph} | \c{alph} } {#1}
660            { \int_set:Nn #2 { \int_from_alph:n {#3} } }
661          \regex_match:nVT { \c{Roman} | \c{roman} } {#1}
662            { \int_set:Nn #2  { \int_from_roman:n {#3} } }
663        }
664    }
665 \cs_generate_variant:Nn \__enumext_start_from:NNn { ccn }
```

(*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.

```
666  \cs_new_protected:Npn \__enumext_widest_from:nNNn #1 #2 #3 #4
667    {
668      \__enumext_if_is_int:nTF {#4}
669        {
670          \setcounter{enumX#1} { #4 }
671        }
672        {
673          \regex_match:nVT { \c{Alph} | \c{alph} } {#2}
674            { \setcounter{enumX#1} { \int_from_alph:n {#4} } }
675          \regex_match:nVT { \c{Roman} | \c{roman} } {#2}
676            { \setcounter{enumX#1} { \int_from_roman:n {#4} } }
677        }
678      \__enumext_label_width_by_box:cv
679        { l__enumext_labelwidth_#1_dim } { l__enumext_label_#1_tl }
680    }
681  \cs_generate_variant:Nn \__enumext_widest_from:nNNn { nccn }
```

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

start
widest
\l__enumext_start_X_int

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

```
682  \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
683    {
684      \keys_define:nn { enumext / #1 }
685        {
686          start  .code:n     = {
687                                  \__enumext_start_from:ccn
688                                    { l__enumext_label_#2_tl }
689                                    { l__enumext_start_#2_int } {##1}
690                                },
691          start  .initial:n = 1,
692          widest .code:n     = {
693                                  \__enumext_widest_from:nccn {#2}
694                                    { l__enumext_label_#2_tl }
695                                    { l__enumext_labelwidth_#2_dim } {##1}
696                                },
697          widest .value_required:n = true,
698          start  .value_required:n = true,
699        }
700    }
701  \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }
```

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

10.13   Setting keys for vertical spaces

topsep
partopsep
parsep
noitemsep
nosep

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

```
702  \cs_set_protected:Npn \__enumext_tmp:nnnnnn #1 #2 #3 #4 #5 #6
703    {
704      \keys_define:nn { enumext / #1 }
705        {
706          topsep    .skip_set:c = { l__enumext_topsep_#2_skip },
707          topsep    .initial:n  = {#3},
708          topsep    .value_required:n = true,
709          partopsep .skip_set:c = { l__enumext_partopsep_#2_skip },
710          partopsep .initial:n  = {#4},
711          partopsep .value_required:n = true,
712          parsep    .skip_set:c = { l__enumext_parsep_#2_skip },
713          parsep    .initial:n  = {#5},
```

```
714        parsep    .value_required:n = true,
715        itemsep   .skip_set:c = { l__enumext_itemsep_#2_skip },
716        itemsep   .initial:n  = {#6},
717        itemsep   .value_required:n = true,
718        noitemsep .meta:n      = { itemsep = 0pt, parsep = 0pt },
719        noitemsep .value_forbidden:n = true,
720        nosep     .meta:n      = {
721                                  itemsep = 0pt, parsep= 0pt,
722                                  topsep = 0pt, partopsep = 0pt,
723                                },
724        nosep     .value_forbidden:n = true,
725      }
726    }
```

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

```
727 \__enumext_tmp:nnnnnn { level-1 } { i } { 8.0pt plus 2.0pt minus 4.0pt }
728    { 2.0pt plus 1.0pt minus 1.0pt } { 4.0pt plus 2.0pt minus 1.0pt }
729    { 4.0pt plus 2.0pt minus 1.0pt }
730 \__enumext_tmp:nnnnnn { level-2 } { ii } { 4.0pt plus 2.0pt minus 1.0pt }
731    { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
732    { 2.0pt plus 1.0pt minus 1.0pt }
733 \__enumext_tmp:nnnnnn { level-3 } { iii } { 2.0pt plus 1.0pt minus 1.0pt }
734    { 1.0pt minus 1.0pt }{ 0pt }{ 2.0pt plus 1.0pt minus 1.0pt }
735 \__enumext_tmp:nnnnnn { level-4 } { iv } { 2.0pt plus 1.0pt minus 1.0pt }
736    { 1.0pt minus 1.0pt }{ 0pt }{ 2.0pt plus 1.0pt minus 1.0pt }
737 \__enumext_tmp:nnnnnn { keyans  } { v }{ 4.0pt plus 2.0pt minus 1.0pt }
738    { 2.0pt plus 1.0pt minus 1.0pt }{ 2.0pt plus 1.0pt minus 1.0pt }
739    { 2.0pt plus 1.0pt minus 1.0pt }
740 \__enumext_tmp:nnnnnn { enumext* } { vii } { 8.0pt plus 2.0pt minus 4.0pt }
741    { 2.0pt plus 1.0pt minus 1.0pt } { 4.0pt plus 2.0pt minus 1.0pt }
742    { 4.0pt plus 2.0pt minus 1.0pt }
743 \__enumext_tmp:nnnnnn { keyans* } { viii } { 4.0pt plus 2.0pt minus 1.0pt }
744    { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
745    { 2.0pt plus 1.0pt minus 1.0pt }
```

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

10.14   Setting keys for horizontal spaces

itemindent
rightmargin
listparindent
list-offset
list-indent

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

```
746 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
747   {
748     \keys_define:nn { enumext / #1 }
749       {
750         itemindent    .dim_set:c = { l__enumext_fake_item_indent_#2_dim },
751         itemindent    .value_required:n = true,
752         rightmargin   .dim_set:c = { l__enumext_rightmargin_#2_dim },
753         rightmargin   .value_required:n = true,
754         listparindent .dim_set:c = { l__enumext_listparindent_#2_dim },
755         listparindent .value_required:n = true,
756         list-offset   .dim_set:c = { l__enumext_listoffset_#2_dim },
757         list-offset   .value_required:n = true,
758         list-indent   .code:n     =
759                         \bool_set_true:c { l__enumext_leftmargin_tmp_#2_bool }
760                         \dim_set:cn { l__enumext_leftmargin_tmp_#2_dim } {##1},
761         list-indent   .value_required:n = true,
762       }
763   }
764 \clist_map_inline:Nn \c__enumext_all_envs_clist { \__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.

```
765 \cs_set_protected:Npn \__enumext_tmp:n #1
766   {
767     \keys_define:nn { enumext / #1 } { list-indent .initial:n  = 0pt, }
768   }
769 \clist_map_inline:nn { enumext*, keyans* } { \__enumext_tmp:n {#1} }
```

10.14.1 Functions for setting the fake `itemindent`

__enumext_fake_item:
__enumext_keyans_fake_item:
__enumext_fake_item_vii:
__enumext_fake_item_viii:

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

```
770 \cs_set_protected:Nn \__enumext_fake_item:
771   {
772     \dim_compare:nNnT
773       { \dim_use:c { l__enumext_fake_item_indent_ \__enumext_level: _dim } }
774       >
775       { \c_zero_dim }
776       {
777         \tl_set:ce { l__enumext_fake_item_indent_ \__enumext_level: _tl }
778           {
779             \exp_not:N \mode_leave_vertical:
780             \exp_not:n { \skip_horizontal:n }
781               { \dim_use:c { l__enumext_fake_item_indent_ \__enumext_level: _dim } }
782             \ignorespaces
783           }
784       }
785   }
786 \cs_set_protected:Nn \__enumext_keyans_fake_item:
787   {
788     \dim_compare:nNnT
789       { \l__enumext_fake_item_indent_v_dim } > { \c_zero_dim }
790       {
791         \tl_set:Ne \l__enumext_fake_item_indent_v_tl
792           {
793             \exp_not:N \mode_leave_vertical:
794             \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_v_dim
795           }
796       }
797   }
798 \cs_set_protected:Nn \__enumext_fake_item_vii:
799   {
800     \dim_compare:nNnT
801       { \l__enumext_fake_item_indent_vii_dim } > { \c_zero_dim }
802       {
803         \tl_set:Ne \l__enumext_fake_item_indent_vii_tl
804           {
805             \exp_not:N \mode_leave_vertical:
806             \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_vii_dim
807           }
808       }
809   }
810 \cs_set_protected:Nn \__enumext_fake_item_viii:
811   {
812     \dim_compare:nNnT
813       { \l__enumext_fake_item_indent_viii_dim } > { \c_zero_dim }
814       {
815         \tl_set:Ne \l__enumext_fake_item_indent_viii_tl
816           {
817             \exp_not:N \mode_leave_vertical:
818             \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_viii_dim
819           }
820       }
821   }
```

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

10.15 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.

```
822 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
823   {
824     \keys_define:nn { enumext / #1 }
825       {
826         show-length .bool_set:c = { l__enumext_show_length_#2_bool },
```

```
827          show-length .initial:n  = false,
828        }
829    }
830  \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }
```

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

10.16 Setting before, after and first keys

before
before*
after
first

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

```
831  \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
832    {
833      \keys_define:nn { enumext / #1 }
834        {
835          before  .tl_set:c   = { l__enumext_before_no_starred_key_#2_tl },
836          before  .value_required:n = true,
837          before* .tl_set:c   = { l__enumext_before_starred_key_#2_tl },
838          before* .value_required:n = true,
839          after   .tl_set:c   = { l__enumext_after_stop_list_#2_tl },
840          after   .value_required:n = true,
841          first   .tl_set:c   = { l__enumext_after_list_args_#2_tl },
842          first   .value_required:n = true,
843        }
844    }
845  \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }
```

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

10.16.1 Functions for before, after and first keys in enumext

__enumext_before_args_exec:
__enumext_before_keys_exec:
__enumext_after_stop_list:
__enumext_after_args_exec:

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

```
846  \cs_new_protected:Nn \__enumext_before_args_exec:
847    {
848      \tl_use:c { l__enumext_before_starred_key_ \__enumext_level: _tl }
849    }
```

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*⟩.

```
850  \cs_new_protected:Nn \__enumext_before_keys_exec:
851    {
852      \tl_use:c { l__enumext_before_no_starred_key_ \__enumext_level: _tl }
853    }
```

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

```
854  \cs_new_protected:Nn \__enumext_after_stop_list:
855    {
856      \tl_use:c { l__enumext_after_stop_list_ \__enumext_level: _tl }
857    }
```

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.

```
858  \cs_new_protected:Nn \__enumext_after_args_exec:
859    {
860      \tl_use:c { l__enumext_after_list_args_ \__enumext_level: _tl }
861    }
```

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

10.16.2 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:

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

```
862  \cs_new_protected:Nn \__enumext_before_args_exec_v:
863    {
864      \tl_use:N \l__enumext_before_starred_key_v_tl
865    }
```

The function __enumext_before_keys_exec_v: executes the {⟨*code*⟩} set by the before key *"before"* the keyans environment is started in {⟨*arg two*⟩} of the list. The {⟨*code*⟩} is executed *"knowing"* all definition and values provides by ⟨*keys*⟩.

```
866 \cs_new_protected:Nn \__enumext_before_keys_exec_v:
867   {
868     \tl_use:N \l__enumext_before_no_starred_key_v_tl
869   }
```

The function __enumext_after_stop_list_v: executes the {⟨*code*⟩} set by the after key *"after"* the keyans environment has finished.

```
870 \cs_new_protected:Nn \__enumext_after_stop_list_v:
871   {
872     \tl_use:N \l__enumext_after_stop_list_v_tl
873   }
```

The function __enumext_after_args_exec_v: executes the {⟨*code*⟩} set by the first key after the end of {⟨*arg two*⟩} of the list defining the keyans environment, just before the first occurrence of \item.

```
874 \cs_new_protected:Nn \__enumext_after_args_exec_v:
875   {
876     \tl_use:N \l__enumext_after_list_args_v_tl
877   }
```

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

10.16.3 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:

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

```
878 \cs_new_protected:Nn \__enumext_before_args_exec_vii:
879   {
880     \tl_use:N \l__enumext_before_starred_key_vii_tl
881   }
882 \cs_new_protected:Nn \__enumext_before_args_exec_viii:
883   {
884     \tl_use:N \l__enumext_before_starred_key_viii_tl
885   }
```

The functions __enumext_before_keys_exec_vii: and __enumext_before_keys_exec_viii: executes the {⟨*code*⟩} set by the before key *"before"* in enumext* and keyans* environments is started in {⟨*arg two*⟩} of the list. The {⟨*code*⟩} is executed *"knowing"* all definition and values provides by ⟨*keys*⟩.

```
886 \cs_new_protected:Nn \__enumext_before_keys_exec_vii:
887   {
888     \tl_use:N \l__enumext_before_no_starred_key_vii_tl
889   }
890 \cs_new_protected:Nn \__enumext_before_keys_exec_viii:
891   {
892     \tl_use:N \l__enumext_before_no_starred_key_viii_tl
893   }
```

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

```
894 \cs_new_protected:Nn \__enumext_after_stop_list_vii:
895   {
896     \tl_use:N \l__enumext_after_stop_list_vii_tl
897   }
898 \cs_new_protected:Nn \__enumext_after_stop_list_viii:
899   {
900     \tl_use:N \l__enumext_after_stop_list_viii_tl
901   }
```

The function __enumext_after_args_exec_v: executes the {⟨*code*⟩} set by the first key after the end of {⟨*arg two*⟩} of the list defining the keyans environment, just before the first occurrence of \item.

```
902 \cs_new_protected:Nn \__enumext_after_args_exec_vii:
903   {
904     \tl_use:N \l__enumext_after_list_args_vii_tl
905   }
906 \cs_new_protected:Nn \__enumext_after_args_exec_viii:
907   {
908     \tl_use:N \l__enumext_after_list_args_viii_tl
909   }
```

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

10.17 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` and `keyans` environments.

```
910  \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
911    {
912      \keys_define:nn { enumext / #1 }
913        {
914          mini-env    .dim_set:c  = { l__enumext_minipage_right_#2_dim },
915          mini-env    .value_required:n = true,
916          mini-sep    .dim_set:c  = { l__enumext_minipage_hsep_#2_dim },
917          mini-sep    .initial:n  = 0.3333em,
918          mini-sep    .value_required:n = true,
919          columns-sep .dim_set:c  = { l__enumext_columns_sep_#2_dim },
920          columns-sep .value_required:n = true,
921          columns     .int_set:c  = { l__enumext_columns_#2_int },
922          columns     .initial:n  = 1,
923          columns     .value_required:n = true,
924        }
925    }
926  \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 default value for `columns` key are 2 and the command `\miniright` is not available, so we will add the keys `miniright` and `miniright*` to implement support for `minipage`.

```
927  \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
928    {
929      \keys_define:nn { enumext / #1 }
930        {
931          columns    .initial:n = 2,
932          miniright  .tl_gset:c = { g__enumext_miniright_code_#2_tl },
933          miniright  .value_required:n = true,
934          miniright* .code:n    = {
935                                   \bool_gset_true:c { g__enumext_minipage_center_#2_bool }
936                                   \keys_set:nn { enumext / #1 } { miniright = {##1} }
937                                  },
938          miniright* .value_required:n = true,
939        }
940    }
941  \clist_map_inline:nn { {enumext*}{vii}, {keyans*}{viii} } { \__enumext_tmp:nn #1 }
```

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

10.18 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.
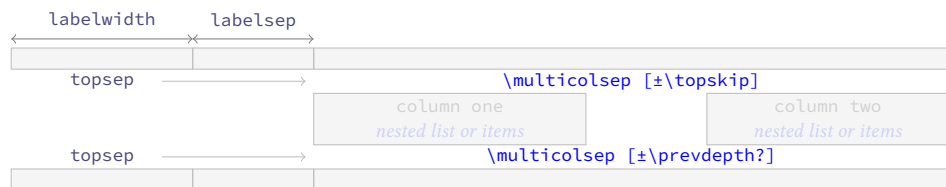
Figure 7: Representation of the vertical space in `multicols` for a nested level.

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.

10.18.1 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*.

```
942  \cs_new_protected:Nn \__enumext_multi_set_vskip:
943    {
944      \skip_set:cn { l__enumext_multicols_above_ \__enumext_level: _skip }
945        {
946          \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
947        }
948      \skip_set:cn { l__enumext_multicols_below_ \__enumext_level: _skip }
949        {
950          \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
951        }
952      \__enumext_add_pre_parsep:
953    }
```

(*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*.

```
954  \cs_new_protected:Nn \__enumext_add_pre_parsep:
955    {
956      \int_case:nn { \l__enumext_level_int }
957        {
958          { 2 }{
959                  \skip_if_eq:nnF { \l__enumext_parsep_i_skip } { \c_zero_skip }
960                    {
961                      \skip_add:Nn \l__enumext_multicols_above_ii_skip { \l__enumext_parsep_i_skip }
962                    }
963                }
964          { 3 }{
965                  \skip_if_eq:nnF { \l__enumext_parsep_ii_skip } { \c_zero_skip }
966                    {
967                      \skip_add:Nn \l__enumext_multicols_above_iii_skip { \l__enumext_parsep_ii_skip
968                    }
969                }
970          { 4 }{
971                  \skip_if_eq:nnF { \l__enumext_parsep_iii_skip } { \c_zero_skip }
972                    {
973                      \skip_add:Nn \l__enumext_multicols_above_iv_skip { \l__enumext_parsep_iii_skip
974                    }
975                }
976        }
977    }
```

(*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*⟩.

```
978  \cs_new_protected:Nn \__enumext_multi_addvspace:
979    {
980      \__enumext_multi_set_vskip:
981      \mode_if_vertical:T
982        {
983          \skip_add:cn { l__enumext_multicols_above_ \__enumext_level: _skip }
984            {
985              \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
986            }
987          \skip_add:cn { l__enumext_multicols_below_ \__enumext_level: _skip }
988            {
989              \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
990            }
991        }
```

```
992        \par\nopagebreak
993        \addvspace{ \skip_use:c { l__enumext_multicols_above_ \__enumext_level: _skip } }
994      }
```

(*End of definition for* __enumext_multi_addvspace:.)

10.18.2   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.

```
995  \cs_new_protected:Nn \__enumext_keyans_multi_set_vskip:
996    {
997      \skip_set:Nn \l__enumext_multicols_above_v_skip
998        {
999          \l__enumext_topsep_v_skip
1000        }
1001      \skip_set:Nn \l__enumext_multicols_below_v_skip
1002        {
1003          \l__enumext_topsep_v_skip
1004        }
1005    }
1006  \cs_new_protected:Nn \__enumext_keyans_multi_addvspace:
1007    {
1008      \__enumext_keyans_multi_set_vskip:
1009      \mode_if_vertical:T
1010        {
1011          \skip_add:Nn \l__enumext_multicols_above_v_skip
1012            {
1013              \skip_use:N \l__enumext_partopsep_v_skip
1014            }
1015          \skip_add:Nn \l__enumext_multicols_below_v_skip
1016            {
1017              \skip_use:N \l__enumext_partopsep_v_skip
1018            }
1019        }
1020      \par\nopagebreak
1021      \addvspace{ \l__enumext_multicols_above_v_skip }
1022    }
```

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

10.19   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.
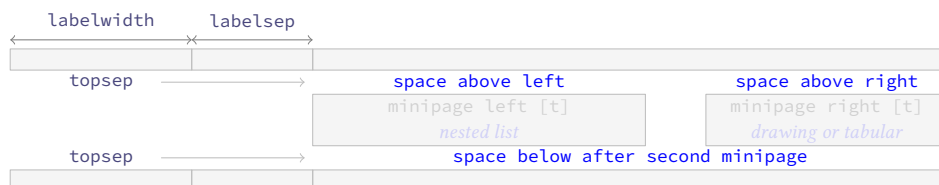
Figure 8: Representation of the minipage spacing adjustment for a nested level.

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[12] package, again my attempts to find the correct values using \showoutput and \showboxdepth absolutely failed.

__enumext_mini_env*

Creates a __enumext_mini_env* 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 \vspace{0pt} to maintain alignment on *"top"*. 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.

```
1023  \DeclareDocumentEnvironment{__enumext_mini_env*}{ m }
1024    {
1025      \__enumext_minipage:w [ t ] { #1 }
1026        \legacy_if_gset_false:n { @minipage }
1027        \vspace { 0pt }
1028    }
1029    { \__enumext_endminipage: }
```

(*End of definition for* __enumext_mini_env*.)

10.19.1   Adjustment of vertical spaces for minipage in enumext

__enumext_mini_set_vskip: The function __enumext_mini_set_vskip: will take care of determining the *"adjust"* spaces that we will apply *"above"* and *"below"* the __enumext_mini_env* 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.

First determine if the multicols environment is active by comparing the value of the \l__enumext_columns_X_int variable handled by the columns key, according to this comparison we set the adjusted values for \l__enumext_minipage_left_skip, \l__enumext_minipage_right_skip and \l__enumext_minipage_after_skip.

```
1030  \cs_new_protected:Nn \__enumext_mini_set_vskip:
1031    {
1032      \int_compare:nNnTF
1033        { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } > { 1 }
1034        {
```

If multicols environment is nested in __enumext_mini_env* environment, we will apply a correction factor to the *vertical spaces* taking into account the value of \topsep of the current level and the value of \parsep of the previous level, if these are zero we will use \strutbox as the basis for the calculations.

```
1035          \skip_if_eq:nnTF
1036            { \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip } } { \c_zero_skip }
1037            {
1038              \skip_set:Nn \l__enumext_minipage_left_skip
1039                {
1040                  -0.150\box_dp:N \strutbox
1041                }
1042              \skip_set:Nn \l__enumext_minipage_right_skip
1043                {
1044                  0.695\box_dp:N \strutbox
1045                }
1046              \skip_set:Nn \l__enumext_minipage_after_skip
1047                {
1048                  \box_dp:N \strutbox
1049                }
1050              \__enumext_zero_parsep:
1051            }
1052            {
1053              \skip_set:Nn \l__enumext_minipage_left_skip
1054                {
1055                  \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
1056                }
1057              \skip_set:Nn \l__enumext_minipage_right_skip
1058                {
1059                  0.695\box_dp:N \strutbox
1060                }
1061              \skip_set:Nn \l__enumext_minipage_after_skip
1062                {
1063                  1.85\box_dp:N \strutbox
1064                  + \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
1065                }
1066            }
1067          }
1068          {
```

If only enumext environment is nested in __enumext_mini_env* environment, we will apply a correction factor to the *vertical spaces* taking into account the value of \topsep, if this is zero we will use \strutbox as the basis for the calculations.

```
1069          \skip_if_eq:nnTF
1070            { \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip } } { \c_zero_skip }
1071            {
1072              \skip_set:Nn \l__enumext_minipage_left_skip
```

```
1073            {
1074                0.5\box_dp:N \strutbox
1075                - \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
1076            }
1077          \skip_set:Nn \l__enumext_minipage_right_skip
1078            {
1079                \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
1080            }
1081          \skip_set:Nn \l__enumext_minipage_after_skip
1082            {
1083                1.6\box_dp:N \strutbox
1084            }
1085        }
1086        {
1087          \skip_set:Nn \l__enumext_minipage_left_skip
1088            {
1089                0.5875\box_dp:N \strutbox
1090                - \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
1091            }
1092          \skip_set:Nn \l__enumext_minipage_right_skip
1093            {
1094                + \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
1095                + \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
1096            }
1097          \skip_set:Nn \l__enumext_minipage_after_skip
1098            {
1099                0.325\box_dp:N \strutbox
1100                + \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
1101            }
1102        }
1103      }
1104    }
```

(*End of definition for __enumext_mini_set_vskip:.*)

__enumext_zero_parsep: The function __enumext_zero_parsep: *"adjusted"* the value of \l__enumext_minipage_after_-
skip detecting the value of \parsep from the previous level. This is necessary since \parsep from the
previous level affects the *vertical spaces* and this is noticeable when using the nosep or noitemsep keys.

```
1105  \cs_new_protected:Nn \__enumext_zero_parsep:
1106    {
1107      \int_case:nn { \l__enumext_level_int }
1108        {
1109          { 2 }{
1110            \skip_if_eq:nnF { \l__enumext_parsep_i_skip } { \c_zero_skip }
1111              {
1112                \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
1113              }
1114          }
1115          { 3 }{
1116            \skip_if_eq:nnF { \l__enumext_parsep_ii_skip } { \c_zero_skip }
1117              {
1118                \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
1119              }
1120          }
1121          { 4 }{
1122            \skip_if_eq:nnF { \l__enumext_parsep_iii_skip } { \c_zero_skip }
1123              {
1124                \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
1125              }
1126          }
1127        }
1128    }
```

(*End of definition for __enumext_zero_parsep:.*)

__enumext_mini_addvspace: The function __enumext_mini_addvspace: will apply the spaces set using \addvspace *"above"* the
__enumext_mini_env* environment in enumext, taking into account whether TeX is in ⟨*horizontal mode*⟩
or ⟨*vertical mode*⟩. For the latter we will make some adjustments since the \partopsep parameter comes
into play and this affects the *vertical spacing*.

```
1129  \cs_new_protected:Nn \__enumext_mini_addvspace:
1130    {
```

```
1131    \__enumext_mini_set_vskip:
1132    \mode_if_vertical:T
1133      {
1134        \skip_add:Nn \l__enumext_minipage_left_skip
1135          {
1136            \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
1137          }
1138        \skip_add:Nn \l__enumext_minipage_after_skip
1139          {
1140            \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
1141          }
1142      }
1143    \par\nopagebreak
1144    \addvspace { \l__enumext_minipage_left_skip }
1145  }
```

(*End of definition for* __enumext_mini_addvspace:.)

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

__enumext_keyans_mini_set_vskip:

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_env* environment in keyans. The implementation of this function is the same as the one used in enumext.

```
1146  \cs_new_protected:Nn \__enumext_keyans_mini_set_vskip:
1147    {
1148      \skip_zero_new:N \l__enumext_minipage_after_skip
1149      \skip_zero_new:N \l__enumext_minipage_left_skip
1150      \skip_zero_new:N \l__enumext_minipage_right_skip
1151      \int_compare:nNnTF { \l__enumext_columns_v_int } > { 1 }
1152        {
1153          \skip_if_eq:nnTF { \l__enumext_topsep_v_skip } { \c_zero_skip }
1154            {
1155              \skip_set:Nn \l__enumext_minipage_left_skip { -0.25\box_dp:N \strutbox }
1156              \skip_set:Nn \l__enumext_minipage_right_skip { 0.705\box_dp:N \strutbox }
1157              \skip_set:Nn \l__enumext_minipage_after_skip { \box_dp:N \strutbox }
1158              \skip_if_eq:nnF { \l__enumext_parsep_i_skip } { \c_zero_skip }
1159                {
1160                  \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
1161                }
1162            }
1163            {
1164              \skip_set:Nn \l__enumext_minipage_left_skip
1165                {
1166                  \skip_use:N \l__enumext_topsep_v_skip
1167                }
1168              \skip_set:Nn \l__enumext_minipage_right_skip
1169                {
1170                  0.705\box_dp:N \strutbox
1171                }
1172              \skip_set:Nn \l__enumext_minipage_after_skip
1173                {
1174                  1.85\box_dp:N \strutbox + \l__enumext_topsep_v_skip
1175                }
1176            }
1177        }
1178        {
1179          \skip_if_eq:nnTF { \l__enumext_topsep_v_skip } { \c_zero_skip }
1180            {
1181              \skip_set:Nn \l__enumext_minipage_left_skip
1182                {
1183                  0.5\box_dp:N \strutbox
1184                  + \l__enumext_partopsep_v_skip
1185                }
1186              \skip_set:Nn \l__enumext_minipage_right_skip
1187                {
1188                  \l__enumext_partopsep_v_skip
1189                }
1190              \skip_set:Nn \l__enumext_minipage_after_skip { 1.6\box_dp:N \strutbox }
1191            }
1192            {
1193              \skip_set:Nn \l__enumext_minipage_left_skip
1194                {
```

```
1195                    0.5875\box_dp:N \strutbox - \l__enumext_partopsep_v_skip
1196                  }
1197              \skip_set:Nn \l__enumext_minipage_right_skip
1198                  {
1199                    \l__enumext_topsep_v_skip + \l__enumext_partopsep_v_skip
1200                  }
1201              \skip_set:Nn \l__enumext_minipage_after_skip
1202                  {
1203                    0.325\box_dp:N \strutbox + \l__enumext_topsep_v_skip
1204                  }
1205            }
1206        }
1207    }
```

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

__enumext_keyans_mini_addvspace: The function __enumext_keyans_mini_addvspace: will apply the spaces set using \addvspace *"above"* the __enumext_mini_env* environment in keyans, taking into account whether TeX is in ⟨*horizontal mode*⟩ or ⟨*vertical mode*⟩. For the latter we will make some adjustments since the \partopsep parameter comes into play and this affects the *vertical spacing*. The implementation of this function is the same as the one used in enumext.

```
1208  \cs_new_protected:Nn \__enumext_keyans_mini_addvspace:
1209    {
1210      \__enumext_keyans_mini_set_vskip:
1211      \mode_if_vertical:T
1212        {
1213          \skip_add:Nn \l__enumext_minipage_left_skip
1214            {
1215              \l__enumext_partopsep_v_skip
1216            }
1217          \skip_add:Nn \l__enumext_minipage_after_skip
1218            {
1219              \l__enumext_partopsep_v_skip
1220            }
1221        }
1222      \par\nopagebreak
1223      \addvspace { \l__enumext_minipage_left_skip }
1224    }
```

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

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

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

```
1225  \cs_new_protected:Nn \__enumext_mini_set_vskip_vii:
1226    {
1227      \skip_zero_new:N \l__enumext_minipage_left_skip
1228      \skip_gzero_new:N \g__enumext_minipage_right_skip
1229      \skip_gzero_new:N \g__enumext_minipage_after_skip
1230      \skip_if_eq:nnTF { \l__enumext_topsep_vii_skip } { \c_zero_skip }
1231        {
1232          \skip_set:Nn \l__enumext_minipage_left_skip { 0.5\box_dp:N \strutbox }
1233          \skip_gset:Nn \g__enumext_minipage_right_skip { 0.325\box_dp:N \strutbox }
1234        }
1235        {
1236          \skip_set:Nn \l__enumext_minipage_left_skip { 0.5875\box_dp:N \strutbox }
1237          \skip_gset:Nn \g__enumext_minipage_right_skip
1238            {
1239              \l__enumext_topsep_vii_skip
1240            }
1241          \skip_gset:Nn \g__enumext_minipage_after_skip
1242            {
1243              0.325\box_dp:N \strutbox + \l__enumext_topsep_vii_skip
1244            }
1245        }
1246    }
1247  \cs_new_protected:Nn \__enumext_mini_set_vskip_viii:
1248    {
1249      \skip_zero_new:N \l__enumext_minipage_after_skip
```

```
1250    \skip_zero_new:N \l__enumext_minipage_left_skip
1251    \skip_zero_new:N \l__enumext_minipage_right_skip
1252    \skip_if_eq:nnTF { \l__enumext_topsep_viii_skip } { \c_zero_skip }
1253      {
1254        \skip_set:Nn \l__enumext_minipage_left_skip
1255          {
1256            0.5\box_dp:N \strutbox
1257          }
1258        \skip_set:Nn \l__enumext_minipage_right_skip
1259          {
1260            \l__enumext_partopsep_viii_skip
1261          }
1262        \skip_set:Nn \l__enumext_minipage_after_skip
1263          {
1264            1.6\box_dp:N \strutbox
1265          }
1266      }
1267      {
1268        \skip_set:Nn \l__enumext_minipage_left_skip
1269          {
1270            0.5875\box_dp:N \strutbox
1271          }
1272        \skip_set:Nn \l__enumext_minipage_right_skip
1273          {
1274            \l__enumext_topsep_viii_skip
1275          }
1276        \skip_set:Nn \l__enumext_minipage_after_skip
1277          {
1278            0.325\box_dp:N \strutbox + \l__enumext_topsep_viii_skip
1279          }
1280      }
1281    }
```

(*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_env* environment on the *left side* when the miniright 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.

```
1282  \cs_new_protected:Nn \__enumext_mini_addvspace_vii:
1283    {
1284      \__enumext_mini_set_vskip_vii:
1285      \par\nopagebreak
1286      \addvspace { \l__enumext_minipage_left_skip }
1287    }
1288  \cs_new_protected:Nn \__enumext_mini_addvspace_viii:
1289    {
1290      \__enumext_mini_set_vskip_viii:
1291      \par\nopagebreak
1292      \addvspace { \l__enumext_minipage_left_skip }
1293    }
```

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

10.19.4 The command \miniright

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

\miniright

First we will perform some checks to prevent the command from being executed outside the enumext environment or from being executed inside the keyanspic environment, then we call the internal functions for the enumext and keyans environments.

```
1294  \NewDocumentCommand \miniright { s }
1295    {
1296      \int_compare:nNnT { \l__enumext_keyans_pic_level_int } = { 1 }
1297        {
1298          \msg_error:nnn { enumext } { wrong-miniright-place }
```

```
1299              }
1300          \int_compare:nNnT { \l__enumext_level_int } = { 0 }
1301              {
1302                  \msg_error:nnn { enumext } { wrong-miniright-place }
1303              }
1304          \int_compare:nNnTF { \l__enumext_keyans_level_int } = { 1 }
1305              {
1306                  \__enumext_keyans_mini_right_cmd:n {#1}
1307              }
1308              { \__enumext_mini_right_cmd:n {#1} }
1309      }
```

(*End of definition for* \miniright. *This function is documented on page 9.*)

__enumext_mini_right_cmd:n   The function __enumext_mini_right_cmd:n takes as argument the *starred version* '*' 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_env* environment on the *"left side"*, then we open the __enumext_mini_env* 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.

```
1310  \cs_new_protected:Npn \__enumext_mini_right_cmd:n #1
1311      {
1312          \dim_compare:nNnTF
1313              { \dim_use:c { l__enumext_minipage_right_ \__enumext_level: _dim } } > { \c_zero_dim }
1314              {
1315                  \__enumext_multicols_stop:
1316                  \end{__enumext_mini_env*}
1317                  \hfill
1318                  \begin{__enumext_mini_env*}
1319                      { \dim_use:c { l__enumext_minipage_right_ \__enumext_level: _dim } }
1320                      \par\addvspace { \l__enumext_minipage_right_skip }
1321                      \bool_if:nF {#1}
1322                          {
1323                              \centering
1324                          }
1325                      \int_gzero:N \g__enumext_minipage_stat_int
1326                  }
1327              { \msg_error:nnn { enumext } { wrong-miniright-use } }
1328      }
```

(*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 version* '*' 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.

```
1329  \cs_new_protected:Npn \__enumext_keyans_mini_right_cmd:n #1
1330      {
1331          \dim_compare:nNnTF { \l__enumext_minipage_right_v_dim } > { \c_zero_dim }
1332              {
1333                  \__enumext_keyans_multicols_stop:
1334                  \end{__enumext_mini_env*}
1335                  \hfill
1336                  \begin{__enumext_mini_env*}{ \l__enumext_minipage_right_v_dim }
1337                      \par\addvspace { \l__enumext_minipage_right_skip }
1338                      \bool_if:nF {#1}
1339                          {
1340                              \centering
1341                          }
1342                      \int_gzero:N \g__enumext_minipage_stat_int
1343                  }
1344              { \msg_error:nnn { enumext } { wrong-miniright-use } }
1345      }
```

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

10.20 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.

```
1346 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1347   {
1348     \keys_define:nn { enumext / #1 }
1349       {
1350         above  .skip_set:c = { l__enumext_vspace_above_#2_skip },
1351         above  .value_required:n = true,
1352         above* .code:n     = \bool_set_true:c { l__enumext_vspace_a_star_#2_bool }
1353                             \keys_set:nn { enumext / #1 } { above = {##1} },
1354         above* .value_required:n = true,
1355         below  .skip_set:c = { l__enumext_vspace_below_#2_skip },
1356         below  .value_required:n = true,
1357         below* .code:n     = \bool_set_true:c { l__enumext_vspace_b_star_#2_bool }
1358                             \keys_set:nn { enumext / #1 } { below = {##1} },
1359         below* .value_required:n = true,
1360       }
1361   }
1362 \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }
```

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

10.20.1 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.

```
1363 \cs_new_protected:Nn \__enumext_vspace_above:
1364   {
1365     \skip_if_eq:nnF
1366       { \skip_use:c { l__enumext_vspace_above_ \__enumext_level: _skip } } { \c_zero_skip }
1367       {
1368         \bool_if:cTF { l__enumext_vspace_a_star_ \__enumext_level: _bool }
1369           {
1370             \vspace*{ \skip_use:c { l__enumext_vspace_above_ \__enumext_level: _skip } }
1371           }
1372           {
1373             \vspace { \skip_use:c { l__enumext_vspace_above_ \__enumext_level: _skip } }
1374           }
1375       }
1376   }
```

(*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.

```
1377 \cs_new_protected:Nn \__enumext_vspace_below:
1378   {
1379     \skip_if_eq:nnF
1380       { \skip_use:c { l__enumext_vspace_below_ \__enumext_level: _skip } } { \c_zero_skip }
1381       {
1382         \bool_if:cTF { l__enumext_vspace_b_star_ \__enumext_level: _bool }
1383           {
1384             \vspace*{ \skip_use:c { l__enumext_vspace_below_ \__enumext_level: _skip } }
1385           }
1386           {
1387             \vspace { \skip_use:c { l__enumext_vspace_below_ \__enumext_level: _skip } }
1388           }
1389       }
1390   }
```

(*End of definition for __enumext_vspace_below:.*)

10.20.2 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.

```
1391 \cs_new_protected:Nn \__enumext_vspace_above_v:
1392   {
1393     \skip_if_eq:nnF { \l__enumext_vspace_above_v_skip } { \c_zero_skip }
1394       {
1395         \bool_if:NTF \l__enumext_vspace_a_star_v_bool
1396           {
1397             \vspace*{ \l__enumext_vspace_above_v_skip }
1398           }
1399           { \vspace { \l__enumext_vspace_above_v_skip } }
1400       }
1401   }
```

(*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.

```
1402 \cs_new_protected:Nn \__enumext_vspace_below_v:
1403   {
1404     \skip_if_eq:nnF { \l__enumext_vspace_below_v_skip } { \c_zero_skip }
1405       {
1406         \bool_if:NTF \l__enumext_vspace_b_star_v_bool
1407           {
1408             \vspace*{ \l__enumext_vspace_below_v_skip }
1409           }
1410           { \vspace { \l__enumext_vspace_below_v_skip } }
1411       }
1412   }
```

(*End of definition for* __enumext_vspace_below_v:.)

10.20.3 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.

```
1413 \cs_new_protected:Nn \__enumext_vspace_above_vii:
1414   {
1415     \skip_if_eq:nnF { \l__enumext_vspace_above_vii_skip } { \c_zero_skip }
1416       {
1417         \bool_if:NTF \l__enumext_vspace_a_star_vii_bool
1418           {
1419             \vspace*{ \l__enumext_vspace_above_vii_skip }
1420           }
1421           { \vspace { \l__enumext_vspace_above_vii_skip } }
1422       }
1423   }
1424 \cs_new_protected:Nn \__enumext_vspace_above_viii:
1425   {
1426     \skip_if_eq:nnF { \l__enumext_vspace_above_viii_skip } { \c_zero_skip }
1427       {
1428         \bool_if:NTF \l__enumext_vspace_a_star_viii_bool
1429           {
1430             \vspace*{ \l__enumext_vspace_above_viii_skip }
1431           }
1432           { \vspace { \l__enumext_vspace_above_viii_skip } }
1433       }
1434   }
```

(*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.

```
1435 \cs_new_protected:Nn \__enumext_vspace_below_vii:
1436   {
1437     \skip_if_eq:nnF { \l__enumext_vspace_below_vii_skip } { \c_zero_skip }
1438       {
1439         \bool_if:NTF \l__enumext_vspace_b_star_vii_bool
1440           {
1441             \vspace*{ \l__enumext_vspace_below_vii_skip }
```

```
1442                    }
1443                    { \vspace { \l__enumext_vspace_below_vii_skip } }
1444                }
1445            }
1446        \cs_new_protected:Nn \__enumext_vspace_below_viii:
1447            {
1448                \skip_if_eq:nnF { \l__enumext_vspace_below_viii_skip } { \c_zero_skip }
1449                    {
1450                        \bool_if:NTF \l__enumext_vspace_b_star_viii_bool
1451                            {
1452                                \vspace*{ \l__enumext_vspace_below_viii_skip }
1453                            }
1454                            { \vspace { \l__enumext_vspace_below_viii_skip } }
1455                    }
1456            }
```

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

10.21   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*⟩.

series
resume
resume*

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

```
1457    \cs_set_protected:Npn \__enumext_tmp:n #1
1458        {
1459            \keys_define:nn { enumext / #1 }
1460                {
1461                    series  .str_set:N = \l__enumext_series_str,
1462                    series  .value_required:n = true,
1463                    resume  .code:n = \__enumext_resume_series:n {##1},
1464                    resume* .code:n = \__enumext_resume_starred:,
1465                    resume* .value_forbidden:n = true,
1466                }
1467        }
1468    \clist_map_inline:nn { level-1, enumext* } { \__enumext_tmp:n {#1} }
```

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

10.21.1   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 value passed to the environment.

```
1469    \cs_new:Npn \__enumext_filter_series:n #1
1470        {
1471            \use:e
1472                {
1473                    \keyval_parse:NNn
1474                        \__enumext_filter_series_key:n
1475                        \__enumext_filter_series_pair:nn {#1}
1476                }
1477        }
```

The function `\__enumext_filter_series_key:n` will be responsible for filtering the ⟨*keys*⟩ that are passed *"without value"* by excluding the `resume` and `resume*` keys.

```
1478    \cs_new:Npn \__enumext_filter_series_key:n #1
1479        {
1480            \str_case:nnF {#1}
1481                {
1482                    { resume } {}
1483                    { resume* } {}
1484                }
1485                { , { \exp_not:n {#1} } }
1486        }
```

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`, `save-ans` and `save-key` keys.

```
1487    \cs_new:Npn \__enumext_filter_series_pair:nn #1#2
1488        {
1489            \str_case:nnF {#1}
1490                {
1491                    { series } {}
```

```
1492          { resume } {}
1493          { start } {}
1494          { save-ans } {}
1495          { save-key } {}
1496        }
1497      { , { \exp_not:n {#1} } = { \exp_not:n {#2} } }
1498    }
```

(*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 (§10.32) and to the function __enumext_parse_keys_vii:n in the enumext* environment definition (§10.35).

```
1499  \cs_new_protected:Npn \__enumext_parse_series:n #1
1500    {
1501      \str_if_empty:NTF \l__enumext_series_str
1502        {
1503          \bool_if:NF \l__enumext_resume_active_bool
1504            {
1505              \__enumext_resume_last:n {#1}
1506            }
1507        }
1508        {
1509          \tl_gclear_new:c { g__enumext_series_ \l__enumext_series_str _tl }
1510          \tl_gset:ce { g__enumext_series_ \l__enumext_series_str _tl }
1511            { \__enumext_filter_series:n {#1} }
1512          \int_if_exist:cF { g__enumext_series_ \l__enumext_series_str _int }
1513            {
1514              \int_new:c { g__enumext_series_ \l__enumext_series_str _int }
1515            }
1516        }
1517    }
```

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

```
1518  \cs_new_protected:Npn \__enumext_resume_last:n #1
1519    {
1520      \bool_if:NT \l__enumext_standar_first_level_bool
1521        {
1522          \tl_gclear:N \g__enumext_standar_series_tl
1523          \tl_gset:Ne \g__enumext_standar_series_tl { \__enumext_filter_series:n {#1} }
1524        }
1525      \bool_if:NT \l__enumext_starred_first_level_bool
1526        {
1527          \tl_gclear:N \g__enumext_starred_series_tl
1528          \tl_gset:Ne \g__enumext_starred_series_tl { \__enumext_filter_series:n {#1} }
1529        }
1530    }
```

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

10.21.2 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 (§10.32) and the `enumext*` environment definition (§10.35).

```
1531 \cs_new_protected:Nn \__enumext_resume_save_counter:
1532   {
1533     \bool_if:NT \g__enumext_standar_bool
1534       {
1535         \tl_if_empty:NF \l__enumext_series_str
1536           {
1537             \int_gset_eq:cN
1538               { g__enumext_series_ \l__enumext_series_str _int } \value{enumXi}
1539           }
1540         \tl_if_empty:NTF \l__enumext_resume_name_tl
1541           {
1542             \str_if_empty:NT \l__enumext_series_str
1543               {
1544                 \int_gset_eq:NN \g__enumext_resume_int \value{enumXi}
1545               }
1546           }
1547           {
1548             \int_if_exist:cT { g__enumext_series_ \l__enumext_resume_name_tl _int }
1549               {
1550                 \int_gset_eq:cN
1551                   { g__enumext_series_ \l__enumext_resume_name_tl _int } \value{enumXi}
1552               }
1553           }
1554         \int_if_exist:cT { g__enumext_resume_ \l__enumext_store_name_tl _int }
1555           {
1556             \int_gset_eq:cN
1557               { g__enumext_resume_ \l__enumext_store_name_tl _int } \value{enumXi}
1558           }
1559       }
1560     \bool_if:NT \g__enumext_starred_bool
1561       {
1562         \tl_if_empty:NF \l__enumext_series_str
1563           {
1564             \int_gset_eq:cN
1565               { g__enumext_series_ \l__enumext_series_str _int } \value{enumXvii}
1566           }
1567         \tl_if_empty:NTF \l__enumext_resume_name_tl
1568           {
1569             \str_if_empty:NT \l__enumext_series_str
1570               {
1571                 \int_gset_eq:NN \g__enumext_resume_vii_int \value{enumXvii}
1572               }
1573           }
1574           {
1575             \int_if_exist:cT { g__enumext_series_ \l__enumext_resume_name_tl _int }
1576               {
1577                 \int_gset_eq:cN
1578                   { g__enumext_series_ \l__enumext_resume_name_tl _int } \value{enumXvii}
1579               }
1580           }
1581         \int_if_exist:cT { g__enumext_resume_ \l__enumext_store_name_tl _int }
1582           {
1583             \int_gset_eq:cN
1584               { g__enumext_resume_ \l__enumext_store_name_tl _int } \value{enumXvii}
1585           }
1586       }
1587   }
```

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

10.21.3 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 $\backslash$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.

```
1588  \cs_new_protected:Npn \__enumext_resume_series:n #1
1589    {
1590      \tl_if_empty:nTF {#1}
1591        {
1592          \__enumext_resume_counter:n { }
1593        }
1594        {
1595          \tl_if_exist:cTF { g__enumext_series_ \tl_to_str:n {#1} _tl }
1596            {
1597              \__enumext_resume_counter:n {#1}
1598              \bool_if:NT \g__enumext_standar_bool
1599                {
1600                  \keys_set:nv { enumext / level-1 }
1601                    { g__enumext_series_ \tl_to_str:n {#1} _tl }
1602                }
1603              \bool_if:NT \g__enumext_starred_bool
1604                {
1605                  \keys_set:nv { enumext / enumext* }
1606                    { g__enumext_series_ \tl_to_str:n {#1} _tl }
1607                }
1608            }
1609            {
1610              \bool_if:NT \g__enumext_standar_bool
1611                {
1612                  \msg_error:nnn { enumext } { unknown-series } {#1}
1613                }
1614              \bool_if:NT \g__enumext_starred_bool
1615                {
1616                  \msg_error:nnn { enumext } { unknown-series } {#1}
1617                }
1618            }
1619        }
1620    }
```

(*End of definition for* $\backslash$__enumext_resume_series:n.)

$\backslash$__enumext_resume_counter:n
$\backslash$__enumext_resume_counter:
$\backslash$__enumext_resume_counter_series:
$\backslash$__enumext_resume_counter_save_ans:

The function $\backslash$__enumext_resume_counter:n will set the variable $\backslash$l__enumext_resume_active_-bool to true and pass the value of the key resume to the variable $\backslash$l__enumext_series_name_tl which will contain the {⟨*series name*⟩}. If the variable $\backslash$l__enumext_series_name_tl is empty, that is, we are passing the key resume *without value*, we will execute the function $\backslash$__enumext_resume_counter: otherwise, when we pass resume={⟨*series name*⟩} we will execute the function $\backslash$__enumext_resume_-counter_series:, finally we will execute the function $\backslash$__enumext_resume_counter_save_ans: which is associated with the key save-ans.

```
1621  \cs_new_protected:Npn \__enumext_resume_counter:n #1
1622    {
1623      \bool_set_true:N \l__enumext_resume_active_bool
1624      \tl_set:Nn \l__enumext_resume_name_tl {#1}
1625      \tl_if_empty:NTF \l__enumext_resume_name_tl
1626        {
1627          \__enumext_resume_counter:
1628        }
1629        {
1630          \__enumext_resume_counter_series:
1631        }
1632      \__enumext_resume_counter_save_ans:
1633    }
```

The $\backslash$__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.

```
1634  \cs_new_protected:Nn \__enumext_resume_counter:
1635    {
1636      \bool_if:NT \g__enumext_standar_bool
1637        {
1638          \int_gincr:N \g__enumext_resume_int
1639          \int_set_eq:NN \l__enumext_start_i_int \g__enumext_resume_int
1640        }
1641      \bool_if:NT \g__enumext_starred_bool
1642        {
1643          \int_gincr:N \g__enumext_resume_vii_int
```

```
1644            \int_set_eq:NN \l__enumext_start_vii_int \g__enumext_resume_vii_int
1645          }
1646      }
```

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.

```
1647  \cs_new_protected:Nn \__enumext_resume_counter_series:
1648    {
1649      \bool_if:NT \g__enumext_standar_bool
1650        {
1651          \int_set:Nn \l__enumext_start_i_int
1652            {
1653              \int_use:c { g__enumext_series_ \l__enumext_resume_name_tl _int } + 1
1654            }
1655        }
1656      \bool_if:NT \g__enumext_starred_bool
1657        {
1658          \int_set:Nn \l__enumext_start_vii_int
1659            {
1660              \int_use:c { g__enumext_series_ \l__enumext_resume_name_tl _int } + 1
1661            }
1662        }
1663    }
```

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.

```
1664  \cs_new_protected:Nn \__enumext_resume_counter_save_ans:
1665    {
1666      \bool_lazy_and:nnT
1667        { \bool_if_p:N \l__enumext_standar_first_level_bool }
1668        { \bool_if_p:N \l__enumext_store_active_bool }
1669        {
1670          \int_set:Nn \l__enumext_start_i_int
1671            {
1672              \int_use:c { g__enumext_resume_ \l__enumext_store_name_tl _int } + 1
1673            }
1674        }
1675      \bool_lazy_and:nnT
1676        { \bool_if_p:N \l__enumext_starred_first_level_bool }
1677        { \bool_if_p:N \l__enumext_store_active_bool }
1678        {
1679          \int_set:Nn \l__enumext_start_vii_int
1680            {
1681              \int_use:c { g__enumext_resume_ \l__enumext_store_name_tl _int } + 1
1682            }
1683        }
1684    }
```

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

10.21.4   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.

```
1685  \cs_new_protected:Nn \__enumext_resume_starred:
1686    {
1687      \bool_if:NT \g__enumext_standar_bool
1688        {
1689          \tl_if_empty:NF \g__enumext_standar_series_tl
1690            {
1691              \__enumext_resume_counter:n { }
1692              \keys_set:nV { enumext / level-1 } \g__enumext_standar_series_tl
1693            }
1694        }
1695      \bool_if:NT \g__enumext_starred_bool
1696        {
1697          \tl_if_empty:NF \g__enumext_starred_series_tl
1698            {
```

```
1699                \__enumext_resume_counter:n { }
1700                \keys_set:nV { enumext / enumext* } \g__enumext_starred_series_tl
1701              }
1702          }
1703      }
```

(*End of definition for* __enumext_resume_starred:.)

10.22   Setting save-ans key

The key save-ans is directly associated with the keys resume and resume*, this will activate the entire
"*storage system*" in the enumext package.

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

```
1704  \cs_set_protected:Npn \__enumext_tmp:n #1
1705    {
1706      \keys_define:nn { enumext / #1 }
1707        {
1708          save-ans .code:n = \__enumext_storing_set:n {##1},
1709          save-ans .value_required:n = true,
1710        }
1711    }
1712  \clist_map_inline:nn { level-1, enumext* } { \__enumext_tmp:n {#1} }
```

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

10.22.1   Internal functions for save-ans key

__enumext_storing_set:n   The function __enumext_storing_set:n first pass the value of the save-ans key to the variable
__enumext_storing_exec: \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 we proceed
to execute the function __enumext_storing_exec: for enumext and enumext* environments.

```
1713  \cs_new_protected:Npn \__enumext_storing_set:n #1
1714    {
1715      \tl_set:Ne \l__enumext_store_name_tl {#1}
1716      \tl_if_empty:NTF \l__enumext_store_name_tl
1717        {
1718          \bool_if:NT \l__enumext_standar_first_level_bool
1719            {
1720              \msg_error:nnn { enumext } { save-ans-empty } { enumext }
1721            }
1722          \bool_if:NT \l__enumext_starred_first_level_bool
1723            {
1724              \msg_error:nnn { enumext } { save-ans-empty } { enumext* }
1725            }
1726        }
1727        {
1728          \bool_if:NT \l__enumext_standar_first_level_bool
1729            {
1730              \msg_note:nnnV
1731                { enumext } { save-ans-ok } { enumext } \l__enumext_store_name_tl
1732              \__enumext_storing_exec:
1733            }
1734          \bool_if:NT \l__enumext_starred_first_level_bool
1735            {
1736              \msg_note:nnnV
1737                { enumext } { save-ans-ok } { enumext* } \l__enumext_store_name_tl
1738              \__enumext_storing_exec:
1739            }
1740        }
1741    }
```

The function __enumext_storing_exec: will set to true the variable \l__enumext_store_active_-
bool which activates the use of the \anskey command and the keyans, keyans* and keyanspic envi-
ronments and will set to true the variable \l__enumext_store_ans_bool used for checking answers
by the check-ans and no-store keys. 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*.

```
1742  \cs_new_protected:Nn \__enumext_storing_exec:
1743    {
1744      \bool_set_true:N \l__enumext_store_active_bool
```

```
1745        \bool_set_true:N \l__enumext_store_ans_bool
1746        \tl_gset:NV \g__enumext_store_name_tl \l__enumext_store_name_tl
1747        \prop_if_exist:cF { g__enumext_ \l__enumext_store_name_tl _prop }
1748          {
1749            \msg_info:nnV { enumext } { store-prop } \l__enumext_store_name_tl
1750            \prop_new:c { g__enumext_ \l__enumext_store_name_tl _prop }
1751          }
1752        \seq_if_exist:cF { g__enumext_ \l__enumext_store_name_tl _seq }
1753          {
1754            \msg_info:nnV { enumext } { store-seq } \l__enumext_store_name_tl
1755            \seq_new:c { g__enumext_ \l__enumext_store_name_tl _seq }
1756          }
1757        \int_if_exist:cF { g__enumext_resume_ \l__enumext_store_name_tl _int }
1758          {
1759            \msg_info:nnV { enumext } { store-int } \l__enumext_store_name_tl
1760            \int_new:c { g__enumext_resume_ \l__enumext_store_name_tl _int }
1761          }
1762      }
```

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

10.23   The check answer mechanism

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

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

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

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

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

The integer variable associated to the sum of each `\item` and `\item*` in the environment `\g__enumext_-count_item_number_int` must match the integer variable `\g__enumext_count_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.

10.23.1   Setting check-ans key

check-ans
no-store

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

```
1763 \cs_set_protected:Npn \__enumext_tmp:n #1
1764   {
1765     \keys_define:nn { enumext / #1 }
1766       {
1767         check-ans .bool_set:N = \l__enumext_check_ans_bool,
1768         check-ans .initial:n  = false,
1769         check-ans .value_required:n = true,
1770         no-store  .code:n = {
1771                             \bool_set_false:N \l__enumext_store_ans_bool
1772                             \bool_set_false:N \l__enumext_check_ans_bool
1773                            },
1774         no-store  .value_forbidden:n = true,
1775       }
1776   }
1777 \clist_map_inline:nn
```

```
1778    {
1779        level-1, level-2, level-3, level-4, enumext*
1780    }
1781    { \__enumext_tmp:n {#1} }
```

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

10.23.2   Set-up check answer mechanism

__enumext_check_ans:
__enumext_check_ans_level:

The function __enumext_check_ans: will check the state of the variable \l__enumext_check_-ans_bool activated by the key check-ans, if this is *"true"* it will check the variable \l__enumext_-store_name_tl is not empty, that is, the key save-ans is activated, if so it will execute the function __enumext_check_ans_level: and otherwise it will return an appropriate error message.

```
1782    \cs_new_protected:Nn \__enumext_check_ans:
1783        {
1784            \bool_if:NT \l__enumext_check_ans_bool
1785                {
1786                    \tl_if_empty:NTF \l__enumext_store_name_tl
1787                        {
1788                            \bool_if:NT \l__enumext_standar_first_level_bool
1789                                {
1790                                    \msg_error:nnnn { enumext } { need-save-ans }{ check-ans } { enumext }
1791                                }
1792                            \bool_if:NT \l__enumext_starred_first_level_bool
1793                                {
1794                                    \msg_error:nnnn { enumext } { need-save-ans }{ check-ans } { enumext* }
1795                                }
1796                        }
1797                        {
1798                            \__enumext_check_ans_level:
1799                        }
1800                }
1801        }
```

The function __enumext_check_ans_level: will decrement by *"one"* the value of the variable \g__-enumext_count_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.

```
1802    \cs_new_protected:Nn \__enumext_check_ans_level:
1803        {
1804            \int_case:nn { \l__enumext_level_int }
1805                {
1806                    { 1 }{
1807                            \bool_lazy_all:nT
1808                                {
1809                                    { \bool_if_p:N \g__enumext_starred_bool }
1810                                    { \int_compare_p:nNn { \l__enumext_level_h_int } = { 1 } }
1811                                }
1812                                {
1813                                    \int_gdecr:N \g__enumext_count_item_number_int
1814                                }
1815                        }
1816                    { 2 }{
1817                            \int_gdecr:N \g__enumext_count_item_number_int
1818                        }
1819                    { 3 }{
1820                            \int_gdecr:N \g__enumext_count_item_number_int
1821                        }
1822                    { 4 }{
1823                            \int_gdecr:N \g__enumext_count_item_number_int
1824                        }
1825                }
1826            \int_case:nn { \l__enumext_level_h_int }
1827                {
1828                    { 1 }{
1829                            \bool_if:NT \g__enumext_standar_bool
1830                                {
1831                                    \int_gdecr:N \g__enumext_count_item_number_int
1832                                }
1833                        }
1834                }
1835        }
```

(End of definition for __enumext_check_ans: and __enumext_check_ans_level:.)

__enumext_check_ans_to_hook:

The function __enumext_check_ans_to_hook: will *export* the value of the local variable \l__enumext_store_name_tl saving it to the global variable \g__enumext_store_name_tl only if the key check-ans is active.

```
1836 \cs_new_protected:Nn \__enumext_check_ans_to_hook:
1837   {
1838     \bool_lazy_and:nnT
1839       { \bool_if_p:N \l__enumext_check_ans_bool }
1840       { \bool_if_p:N \g__enumext_standar_bool }
1841       {
1842         \tl_gset:NV \g__enumext_store_name_tl \l__enumext_store_name_tl
1843       }
1844     \bool_lazy_and:nnT
1845       { \bool_if_p:N \l__enumext_check_ans_bool }
1846       { \bool_if_p:N \g__enumext_starred_bool }
1847       {
1848         \tl_gset:NV \g__enumext_store_name_tl \l__enumext_store_name_tl
1849       }
1850   }
```

(End of definition for __enumext_check_ans_to_hook:.)

__enumext_check_ans_show:

The function __enumext_check_ans_show: will perform the comparison between the \item in the environments and the \item's with answers and return the appropriate message. As this function is passed to the function __enumext_after_env:nn for the environments enumext and enumext* we must make sure that we are not nested at any level and finally reset our global variables.

```
1851 \cs_new_protected:Nn \__enumext_check_ans_show:
1852   {
1853     \int_compare:nNnT { \l__enumext_level_int } = { 0 }
1854       {
1855         \tl_if_empty:NF \g__enumext_store_name_tl
1856           {
1857             \int_compare:nNnTF
1858               { \g__enumext_count_item_number_int } = { \g__enumext_count_item_anskey_int }
1859               {
1860                 \msg_term:nnV { enumext } { items-same-answer } \g__enumext_store_name_tl
1861               }
1862               {
1863                 \msg_warning:nnVV
1864                   { enumext } { item-different-answer }
1865                   \g__enumext_store_name_tl \g__enumext_check_start_line_env_tl
1866               }
1867           }
1868         \int_gzero:N \g__enumext_count_item_number_int
1869         \int_gzero:N \g__enumext_count_item_anskey_int
1870         \bool_gset_false:N \g__enumext_standar_bool
1871         \bool_gset_false:N \g__enumext_starred_bool
1872         \tl_gclear:N \g__enumext_store_name_tl
1873         \tl_gclear:N \g__enumext_check_start_line_env_tl
1874       }
1875   }
```

(End of definition for __enumext_check_ans_show:.)

10.23.3 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.

```
1876 \cs_new_protected:Npn \__enumext_check_starred_cmd:n #1
1877   {
1878     \int_compare:nNnT
1879       { \g__enumext_check_starred_cmd_int } = { 0 }
1880       {
1881         \msg_warning:nnnV
1882           { enumext } { missing-starred }{ #1 } \l__enumext_check_start_line_env_tl
1883       }
1884     \int_compare:nNnT
1885       { \g__enumext_check_starred_cmd_int } > { 1 }
1886       {
```

```
1887            \msg_warning:nnnV
1888              { enumext } { many-starred }{ #1 } \l__enumext_check_start_line_env_tl
1889            }
1890        \int_gzero:N \g__enumext_check_starred_cmd_int
1891        \tl_clear:N \l__enumext_check_start_line_env_tl
1892      }
```

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

10.24   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*.

```
1893 \cs_set_protected:Npn \__enumext_tmp:n #1
1894   {
1895     \keys_define:nn { enumext / #1 }
1896       {
1897         wrap-ans   .cs_set_protected:Np = \__enumext_anskey_wrapper:n ##1,
1898         wrap-ans   .initial:n = \fbox{##1},
1899         wrap-ans   .value_required:n = true,
1900         wrap-opt   .cs_set_protected:Np = \__enumext_keyans_wrapper_opt:n ##1,
1901         wrap-opt   .initial:n = [{##1}],
1902         wrap-opt   .value_required:n = true,
1903         save-sep   .tl_set:N  = \l__enumext_store_keyans_item_opt_sep_tl,
1904         save-sep   .initial:n = {, ~ },
1905         save-sep   .value_required:n = true,
1906         mark-ans   .tl_set:N  = \l__enumext_mark_answer_sym_tl,
1907         mark-ans   .initial:n = \textasteriskcentered,
1908         mark-ans   .value_required:n = true,
1909         mark-pos   .choice:,
1910         mark-pos  / left   .code:n = \str_set:Nn \l__enumext_mark_position_str { l },
1911         mark-pos  / right  .code:n = \str_set:Nn \l__enumext_mark_position_str { r },
1912         mark-pos   .initial:n = right,
1913         mark-pos   .value_required:n = true,
1914         show-ans   .bool_set:N = \l__enumext_show_answer_bool,
1915         show-ans   .initial:n  = false,
1916         show-ans   .value_required:n = true,
1917         show-pos   .bool_set:N = \l__enumext_show_position_bool,
1918         show-pos   .initial:n  = false,
1919         show-pos   .value_required:n = true,
1920         mark-ref   .tl_set:N   = \l__enumext_mark_ref_sym_tl,
1921         mark-ref   .initial:n  = \textasteriskcentered,
1922         mark-ref   .value_required:n = true,
1923         save-ref   .bool_set:N = \l__enumext_store_ref_key_bool,
1924         save-ref   .initial:n  = false,
1925         save-ref   .value_required:n = true,
1926       }
1927   }
1928 \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.

```
1929 \cs_set_protected:Npn \__enumext_tmp:n #1
1930   {
1931     \keys_define:nn { enumext / #1 }
1932       {
1933         mark-pos .choice:,
1934         mark-pos / left  .code:n = \str_set:Nn \l__enumext_mark_position_str { l },
1935         mark-pos / right .code:n = \str_set:Nn \l__enumext_mark_position_str { r },
1936         mark-pos .initial:n = right,
1937         mark-pos .value_required:n  = true,
1938         show-ans .bool_set:N = \l__enumext_show_answer_bool,
1939         show-ans .initial:n  = false,
1940         show-ans .value_required:n = true,
1941         show-pos .bool_set:N = \l__enumext_show_position_bool,
1942         show-pos .initial:n  = false,
1943         show-pos .value_required:n = true,
1944       }
1945   }
```

```
1946   \clist_map_inline:nn { keyans, keyans* } { \__enumext_tmp:n {#1} }
```

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

columns*
columns-sep*

For the enumext and enumext* environments we will only add the keys columns* and columns-sep*. The values set by these keys will be passed as optional arguments to the *"inner levels"* of the enumext and enumext* environments via the __enumext_store_level_open: function used by the *"storage system"* to preserve the structure and then used by the \printkeyans command.

```
1947   \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1948     {
1949       \keys_define:nn { enumext / #1 }
1950         {
1951           columns*      .code:n = \bool_set_true:c { l__enumext_store_columns_#2_bool }
1952                                   \int_set:cn { l__enumext_store_columns_#2_int } {##1}
1953                                   \tl_put_right:ce { l__enumext_store_opt_#2_tl }
1954                                     {
1955                                       columns = \exp_not:v { l__enumext_store_columns_#2_int },
1956                                     },
1957           columns*      .value_required:n  = true,
1958           columns-sep* .code:n = \bool_set_true:c { l__enumext_store_columns_sep_#2_bool }
1959                                   \dim_set:cn { l__enumext_store_columns_sep_#2_dim } {##1}
1960                                   \tl_put_right:ce { l__enumext_store_opt_#2_tl }
1961                                     {
1962                                       columns-sep = \exp_not:v { l__enumext_store_columns_sep_#2_di
1963                                     },
1964           columns-sep* .value_required:n  = true,
1965         }
1966     }
1967   \clist_map_inline:nn
1968     {
1969       {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {enumext*}{vii}
1970     }
1971     { \__enumext_tmp:nn #1 }
```

(*End of definition for* columns* *and* columns-sep*.*)

10.24.1   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.

```
1972   \cs_new_protected:Npn \__enumext_store_addto_prop:n #1
1973     {
1974       \prop_gput_if_not_in:cen { g__enumext_ \l__enumext_store_name_tl _prop }
1975         {
1976           \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } + 1 }
1977         }
1978         { #1 }
1979     }
1980   \cs_generate_variant:Nn \__enumext_store_addto_prop:n { V }
```

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

10.24.2   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.

```
1981   \cs_new_protected:Npn \__enumext_store_addto_seq:n #1
1982     {
1983       \seq_gput_right:cn { g__enumext_ \l__enumext_store_name_tl _seq } { #1 }
1984     }
1985   \cs_generate_variant:Nn \__enumext_store_addto_seq:n { v, V }
```

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

10.24.3   Functions for storing the list structure in the sequence

__enumext_store_level_open:
__enumext_store_level_close:

The memorization structure of the list is handled by the functions __enumext_store_level_open: and __enumext_store_level_close: which are executed per level within the enumext environment. As this structure will be stored in the sequence set by the save-ans key, we will not be able to modify it locally, so it is better to take only two copies of the values set by the columns and columns-sep keys if they are present when changing levels within the enumext environment when executing \anskey. We will store these values in the variable \l__enumext_store_columns_X_tl if they are different from 0 and 0pt and pass them as an optional argument to the environment stored in the sequence enumext.

```
1986  \cs_new_protected:Nn \__enumext_store_level_open:
1987    {
1988      \bool_if:NT \l__enumext_store_ans_bool
1989        {
1990          \tl_if_empty:cTF { l__enumext_store_opt_ \__enumext_level: _tl }
1991            {
1992              \__enumext_store_addto_seq:n
1993                {
1994                  \item \begin{enumext}
1995                }
1996            }
1997            {
1998              \tl_put_left:cn { l__enumext_store_opt_ \__enumext_level: _tl }
1999                {
2000                  \item \begin{enumext} [
2001                }
2002              \tl_put_right:cn { l__enumext_store_opt_ \__enumext_level: _tl }
2003                {
2004                  ]
2005                }
2006              \__enumext_store_addto_seq:v { l__enumext_store_opt_ \__enumext_level: _tl }
2007            }
2008        }
2009    }
2010  \cs_new_protected:Nn \__enumext_store_level_close:
2011    {
2012      \bool_if:NT \l__enumext_store_ans_bool
2013        {
2014          \__enumext_store_addto_seq:n { \end{enumext} }
2015        }
2016    }
```

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

__enumext_store_level_open_vii:
__enumext_store_level_close_vii:

When nesting the enumext* environment in enumext starting right after \item (without material between them) there is a problem with the alignment of the labels with the baseline between the two environments. One way to get around this problem is to place \mode_leave_vertical: and then apply \vspace taking into account \baselineskip, the value of \parsep of the current level of enumext and the value of \topsep of the enumext* environment.

```
2017  \cs_new_protected:Nn \__enumext_store_level_open_vii:
2018    {
2019      \bool_if:NT \l__enumext_store_ans_bool
2020        {
2021          \tl_if_empty:NTF \l__enumext_store_opt_vii_tl
2022            {
2023              \__enumext_store_addto_seq:n
2024                {
2025                  \item \mode_leave_vertical:
2026                    \vspace { -\skip_eval:n { \baselineskip + \parsep } }
2027                    \begin{enumext*}[before={\setlength{\topsep}{0pt}},]
2028                }
2029            }
2030            {
2031              \tl_put_left:Nn \l__enumext_store_opt_vii_tl
2032                {
2033                  \item \mode_leave_vertical:
2034                    \vspace { -\skip_eval:n { \baselineskip + \parsep } }
2035                    \begin{enumext*}[before={\setlength{\topsep}{0pt}},
2036                }
2037              \tl_put_right:Nn \l__enumext_store_opt_vii_tl
2038                {
```

```
2039                    ]
2040                }
2041            \__enumext_store_addto_seq:V \l__enumext_store_opt_vii_tl
2042        }
2043    }
2044  }
2045 \cs_new_protected:Nn \__enumext_store_level_close_vii:
2046  {
2047    \bool_if:NT \l__enumext_store_ans_bool
2048      {
2049        \__enumext_store_addto_seq:n { \end{enumext*} }
2050      }
2051  }
```

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

10.24.4   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

```
2052 \cs_new_protected:Nn \__enumext_print_keyans_box:NN
2053  {
2054    \mode_leave_vertical:
2055    \skip_horizontal:n { -\dim_use:N #2 }
2056    \makebox[0pt][ r ]
2057      {
2058        \makebox[ \dim_use:N #1 ][ \l__enumext_mark_position_str ]
2059          {
2060            \tl_use:N \l__enumext_mark_answer_sym_tl
2061          }
2062      }
2063    \skip_horizontal:n { \dim_use:N #2 }
2064  }
2065 \cs_generate_variant:Nn \__enumext_print_keyans_box:NN { cc }
```

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

10.25   The command \anskey **and internal label and ref**

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 very similar behaviour to that of \item in the enumext and enumext* environments.

\anskey

We want the command to be executed as follows: \anskey(⟨*number*⟩)*[⟨*key* = *val*⟩]{⟨*content*⟩} so first we'll add the keys item-sym*, item-pos* and store-brk.

```
2066 \keys_define:nn { enumext / anskey }
2067  {
2068    item-sym* .tl_set:N   = \l__enumext_store_item_symbol_tl,
2069    item-sym* .value_required:n = true,
2070    item-pos* .dim_set:N  = \l__enumext_store_item_symbol_sep_dim,
2071    item-pos* .value_required:n = true,
2072    store-brk .bool_set:N = \l__enumext_store_columns_break_bool,
2073    store-brk .default:n  = true,
2074    store-brk .value_forbidden:n = true,
2075  }
```

This command \anskey will only be present when using the save-ans key in enumext and enumext* environments, otherwise it will return an error. If the check-ans key is active, increment \g__enumext_-count_item_with_ans_int, then call internal function __enumext_store_anskey_code:nnnn will *"store content"* in the ⟨*sequence*⟩ and in the ⟨*prop list*⟩.

```
2076 \NewDocumentCommand \anskey { d() s o +m }
2077  {
2078    \bool_if:NF \l__enumext_store_active_bool
2079      {
2080        \msg_error:nnnn { enumext } { anskey-wrong-place }{ anskey }{ enumext }
2081      }
2082    \int_compare:nNnT { \l__enumext_keyans_level_int } = { 1 }
2083      {
```

```
2084        \msg_error:nnnn { enumext } { command-wrong-place }{ anskey }{ keyans }
2085      }
2086    \int_compare:nNnT { \l__enumext_keyans_pic_level_int } = { 1 }
2087      {
2088        \msg_error:nnnn { enumext } { command-wrong-place }{ anskey }{ keyanspic }
2089      }
2090    \group_begin:
2091      \bool_if:NT \l__enumext_store_ans_bool
2092        {
2093          \bool_if:NT \l__enumext_check_ans_bool
2094            {
2095              \int_gincr:N \g__enumext_count_item_anskey_int
2096            }
2097          \__enumext_store_anskey_code:nnnn {#1} {#2} {#3} {#4}
2098        }
2099    \group_end:
2100  }
```

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

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

```
2101 \cs_new_protected:Npn \__enumext_store_anskey_code:nnnn #1 #2 #3 #4
2102  {
2103    \__enumext_store_addto_prop:n {#4}
2104    \bool_if:NT \l__enumext_store_ref_key_bool
2105      {
2106        \__enumext_store_internal_ref:
2107      }
2108    \__enumext_store_anskey_show_left:n { #4 }
```

Now we start processing the optional arguments 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 [⟨*key = val*⟩], if the store-brk key is present and the command is running under enumext (not in the starred version) we will add \columnbreak and then \item.

```
2109    \tl_clear:N \l__enumext_store_anskey_arg_tl
2110    \tl_if_novalue:nF {#3}
2111      {
2112        \keys_set:nn { enumext / anskey } {#3}
2113      }
2114    \bool_lazy_and:nnT
2115      { \bool_if_p:N \l__enumext_store_columns_break_bool }
2116      { \bool_not_p:n { \l__enumext_starred_bool } }
2117      {
2118        \tl_put_left:Nn \l__enumext_store_anskey_arg_tl { \columnbreak }
2119      }
2120    \tl_put_right:Nn \l__enumext_store_anskey_arg_tl { \item }
```

Now we will check the (⟨*number*⟩) argument and add it to \l__enumext_store_anskey_arg_tl if the command is running under enumext* (starred version).

```
2121      \tl_if_novalue:nF {#1}
2122        {
2123          \int_set:Nn \l__enumext_store_columns_join_int {#1}
2124          \bool_if:NT \l__enumext_starred_bool
2125            {
2126              \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2127                {
2128                  ( \exp_not:V \l__enumext_store_columns_join_int )
2129                }
2130            }
2131        }
```

And now we will review the starred argument * together with the keys item-sym* and item-pos* and pass them to \l__enumext_store_anskey_arg_tl.

```
2132      \bool_if:nTF {#2}
2133        {
2134          \tl_put_right:Nn \l__enumext_store_anskey_arg_tl { * }
```

```
2135          \tl_if_empty:NF \l__enumext_store_item_symbol_tl
2136            {
2137              \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2138                {
2139                  [ \exp_not:V \l__enumext_store_item_symbol_tl ]
2140                }
2141            }
2142          \dim_compare:nT
2143            {
2144              \l__enumext_store_item_symbol_sep_dim != \c_zero_dim
2145            }
2146            {
2147              \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2148                {
2149                  [ \exp_not:V \l__enumext_store_item_symbol_sep_dim ]
2150                }
2151            }
2152          \tl_put_right:Nn \l__enumext_store_anskey_arg_tl {#4}
2153        }
2154        {
2155          \tl_put_right:Nn \l__enumext_store_anskey_arg_tl {#4}
2156        }
```

Finally we check 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 in ⟨*sequence*⟩.

```
2157        \bool_lazy_and:nnT
2158          { \bool_if_p:N \l__enumext_store_ref_key_bool }
2159          { \bool_if_p:N \l__enumext_hyperref_bool }
2160          {
2161            \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2162              {
2163                \hfill \exp_not:N \hyperlink { \exp_not:V \l__enumext_newlabel_arg_one_tl }
2164                  { \exp_not:V \l__enumext_mark_ref_sym_tl }
2165              }
2166          }
2167        \__enumext_store_addto_seq:V \l__enumext_store_anskey_arg_tl
2168      }
```

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

`\__enumext_store_internal_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`.

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`.

```
2169  \cs_new_protected:Nn \__enumext_store_internal_ref:
2170    {
2171      \cs_set_protected:Npn \__enumext_tmp:n ##1
2172        {
2173          \tl_set_eq:cc { l__enumext_label_copy_##1_tl } { l__enumext_label_##1_tl }
2174          \tl_reverse:c { l__enumext_label_copy_##1_tl }
2175          \tl_remove_once:cn { l__enumext_label_copy_##1_tl } { . }
2176          \tl_reverse:c { l__enumext_label_copy_##1_tl }
2177        }
2178      \clist_map_inline:nn { i, ii, iii, iv, vii } { \__enumext_tmp:n {##1} }
2179      \cs_set:Npn \__enumext_tmp:n ##1
2180        { . \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` is running alone in it or if it is running in a nested `enumext` environment within the starting environment.

```
2181      \bool_lazy_all:nT
2182        {
2183          { \bool_if_p:N \g__enumext_starred_bool }
2184          { \int_compare_p:nNn { \l__enumext_level_int } = { \c_zero_int } }
2185        }
2186        {
2187          \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2188            { \tl_use:N \l__enumext_label_copy_vii_tl }
2189        }
2190      \bool_lazy_all:nT
2191        {
```

```
2192          { \bool_if_p:N \l__enumext_standar_bool }
2193          { \bool_if_p:N \g__enumext_starred_bool }
2194          { \int_compare_p:nNn { \l__enumext_level_int } > { \c_zero_int } }
2195        }
2196        {
2197          \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2198            {
2199              \tl_use:N \l__enumext_label_copy_vii_tl
2200              \int_step_function:nnN { 1 } { \l__enumext_level_int } \__enumext_tmp:n
2201            }
2202        }
```

If started with enumext and if \anskey is running alone in it or if it is running in a nested enumext* environment within the starting environment.

```
2203        \bool_lazy_all:nT
2204          {
2205            { \bool_if_p:N \l__enumext_standar_bool }
2206            { \int_compare_p:nNn { \l__enumext_level_int } > { \c_zero_int } }
2207            { \int_compare_p:nNn { \l__enumext_level_h_int } = { \c_zero_int } }
2208            { \bool_not_p:n { \l__enumext_starred_bool } }
2209          }
2210          {
2211            \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2212              {
2213                \tl_use:N \l__enumext_label_copy_i_tl
2214                \int_step_function:nnN { 2 } { \l__enumext_level_int } \__enumext_tmp:n
2215              }
2216          }
2217        \cs_set:Npn \__enumext_tmp:n ##1
2218          { \tl_use:c { l__enumext_label_copy_ \int_to_roman:n {##1} _tl } }
2219        \bool_lazy_all:nT
2220          {
2221            { \bool_if_p:N \l__enumext_standar_bool }
2222            { \int_compare_p:nNn { \l__enumext_level_int } > { \c_zero_int } }
2223            { \bool_not_p:n { \g__enumext_starred_bool } }
2224            { \int_compare_p:nNn { \l__enumext_level_h_int } > { \c_zero_int } }
2225          }
2226          {
2227            \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2228              {
2229                \int_step_function:nnN { 1 } { \l__enumext_level_int } \__enumext_tmp:n
2230                . \tl_use:N \l__enumext_label_copy_vii_tl
2231              }
2232          }
```

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

```
2233        \tl_put_right:Ne \l__enumext_newlabel_arg_one_tl
2234          {
2235            \l__enumext_store_name_tl \c_colon_str
2236            \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } }
2237          }
```

Now execute the function __enumext_newlabel:nn and save the result in the variable \l__enumext_store_write_aux_file_tl and finally we write in the .aux file.

```
2238        \tl_put_right:Ne \l__enumext_store_write_aux_file_tl
2239          {
2240            \__enumext_newlabel:nn
2241              { \exp_not:V \l__enumext_newlabel_arg_one_tl }
2242              { \l__enumext_newlabel_arg_two_tl }
2243          }
2244        \l__enumext_store_write_aux_file_tl
2245      }
```

(*End of definition for* __enumext_store_internal_ref:.)

__enumext_store_anskey_show_wrap:n   The function __enumext_store_anskey_show_wrap:n *"wraps"* the ⟨*argument*⟩ passed to \anskey when using the wrap-ans key.

```
2246  \cs_new_protected:Npn \__enumext_store_anskey_show_wrap:n #1
2247    {
2248      \par
2249      \bool_if:NT \l__enumext_starred_bool
```

```
2250        {
2251          \cs_set:Nn \__enumext_level: { vii }
2252        }
2253      \__enumext_print_keyans_box:cc
2254        { l__enumext_labelwidth_ \__enumext_level: _dim }
2255        { l__enumext_labelsep_ \__enumext_level: _dim }
2256      \__enumext_anskey_wrapper:n { #1 }
2257    }
```

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

__enumext_store_anskey_show_left:n   The function __enumext_store_anskey_show_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 on the right side when using the show-ans key.

```
2258  \cs_new_protected:Npn \__enumext_store_anskey_show_left:n #1
2259    {
2260      \bool_if:NT \l__enumext_show_answer_bool
2261        {
2262          \__enumext_store_anskey_show_wrap:n { #1 }
2263        }
2264      \bool_if:NT \l__enumext_show_position_bool
2265        {
2266          \tl_set:Ne \l__enumext_mark_answer_sym_tl
2267            {
2268              \group_begin:
2269              \exp_not:N \normalfont
2270              \exp_not:N \footnotesize [ \int_eval:n
2271                {
2272                  \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
2273                }
2274                ]
2275              \group_end:
2276            }
2277          \__enumext_store_anskey_show_wrap:n { #1 }
2278        }
2279    }
```

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

10.26 Common functions for keyans, keyans* and keyanspic

10.26.1 Storing content in prop list

__enumext_keyans_addto_prop:n   The function __enumext_keyans_addto_prop:n will pass the contents of the current ⟨*label*⟩ \l__enumext_label_v_tl for the keyans environment and the current ⟨*label*⟩ \l__enumext_label_vi_tl for the keyanspic environment when using \item* and \anspic*, followed by the *contents* of the optional argument of both commands to the \l__enumext_store_keyans_label_tl variable, which will be passed to the ⟨*prop list*⟩ defined by the save-ans key using the __enumext_store_addto_-prop:V.

```
2280  \cs_new_protected:Npn \__enumext_keyans_addto_prop:n #1
2281    {
2282      \tl_clear:N \l__enumext_store_keyans_label_tl
2283      \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
2284        {
2285          \tl_put_right:Ne \l__enumext_store_keyans_label_tl { \l__enumext_label_vi_tl }
2286        }
2287        {
2288          \tl_put_right:Ne \l__enumext_store_keyans_label_tl { \l__enumext_label_v_tl }
2289        }
2290      \tl_if_novalue:nF { #1 }
2291        {
2292          % Set save-sep
2293          \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_tl
2294            {
2295              \tl_put_right:Ne \l__enumext_store_keyans_label_tl { \l__enumext_store_keyans_item_op
2296            }
2297          \tl_put_right:Ne \l__enumext_store_keyans_label_tl { #1 }
2298        }
2299      \__enumext_store_addto_prop:V \l__enumext_store_keyans_label_tl
2300    }
```

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

10.26.2   The save-ref key for keyans, keyans* and keyanspic

The internal *"label and ref"* system for the keyans, keyans* and keyanspic environments has slight differences with the one implemented for the \anskey command, basically because in this environments we are interested in the current ⟨*label*⟩. The mechanism defined here will allow to execute \ref{⟨*store name : position*⟩} and will return 1.(A).

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

```
2301  \cs_new_protected:Nn \__enumext_keyans_store_ref:
2302    {
2303      \bool_if:NT \l__enumext_store_ref_key_bool
2304        {
2305          \cs_set_protected:Npn \__enumext_tmp:n ##1
2306            {
2307              \tl_set_eq:cc { l__enumext_label_copy_##1_tl } { l__enumext_label_##1_tl }
2308              \tl_reverse:c { l__enumext_label_copy_##1_tl }
2309              \tl_remove_once:cn { l__enumext_label_copy_##1_tl } { . }
2310              \tl_reverse:c { l__enumext_label_copy_##1_tl }
2311            }
2312          \clist_map_inline:nn { i, v, vi, vii, viii } { \__enumext_tmp:n {##1} }
2313          \__enumext_keyans_store_ref_aux_i:
2314        }
2315    }
```

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.

```
2316  \cs_new_protected:Nn \__enumext_keyans_store_ref_aux_i:
2317    {
2318      \bool_if:NT \g__enumext_starred_bool
2319        {
2320          \tl_set_eq:NN \l__enumext_label_copy_i_tl \l__enumext_label_copy_vii_tl
2321        }
2322      \int_compare:nNnT { \l__enumext_keyans_pic_level_int } = { 1 }
2323        {
2324          \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2325            { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_vi_tl }
2326        }
2327      \int_compare:nNnT { \l__enumext_keyans_level_int } = { 1 }
2328        {
2329          \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2330            { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_v_tl }
2331        }
2332      \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
2333        {
2334          \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2335            { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_viii_tl }
2336        }
2337      \tl_put_right:Ne \l__enumext_newlabel_arg_one_tl
2338        {
2339          \l__enumext_store_name_tl \c_colon_str
2340          \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } }
2341        }
2342      \__enumext_keyans_store_ref_aux_ii:
2343    }
```

Now auxiliary function __enumext_keyans_store_ref_aux_ii: save the result in the variable \l__-
enumext_store_write_aux_file_tl and finally we write in the .aux file.

```
2344  \cs_new_protected:Nn \__enumext_keyans_store_ref_aux_ii:
2345    {
2346      \tl_put_right:Ne \l__enumext_store_write_aux_file_tl
2347        {
2348          \__enumext_newlabel:nn
2349            { \exp_not:V \l__enumext_newlabel_arg_one_tl }
2350            { \l__enumext_newlabel_arg_two_tl }
2351        }
2352      \l__enumext_store_write_aux_file_tl
2353    }
```

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

10.26.3 Storing content in sequence

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

```
2354  \cs_new_protected:Npn \__enumext_keyans_addto_seq:n #1
2355    {
2356      \tl_clear:N \l__enumext_store_keyans_label_tl
2357      \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
2358        {
2359          \tl_put_right:Ne \l__enumext_store_keyans_label_tl { \item \l__enumext_label_vi_tl }
2360        }
2361        {
2362          \tl_put_right:Ne \l__enumext_store_keyans_label_tl { \item \l__enumext_label_v_tl }
2363        }
2364      \tl_if_novalue:nF { #1 }
2365        {
2366          \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_tl
2367            {
2368              \tl_put_right:Ne \l__enumext_store_keyans_label_tl
2369                {
2370                  \l__enumext_store_keyans_item_opt_sep_tl
2371                }
2372            }
2373          \tl_put_right:Ne \l__enumext_store_keyans_label_tl { #1 }
2374        }
2375      \__enumext_keyans_addto_seq_link:
2376    }
```

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_keyans_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_count_item_anskey_int` handled by the `check-ans` key.

```
2377  \cs_new_protected:Nn \__enumext_keyans_addto_seq_link:
2378    {
2379      \bool_lazy_and:nnT
2380        { \bool_if_p:N \l__enumext_store_ref_key_bool }
2381        { \bool_if_p:N \l__enumext_hyperref_bool }
2382        {
2383          \tl_put_right:Ne \l__enumext_store_keyans_label_tl
2384            {
2385              \hfill \exp_not:N \hyperlink
2386                {
2387                  \exp_not:V \l__enumext_newlabel_arg_one_tl
2388                }
2389                { \exp_not:V \l__enumext_mark_ref_sym_tl }
2390            }
2391        }
2392      \__enumext_store_addto_seq:V \l__enumext_store_keyans_label_tl
2393      \bool_if:NT \l__enumext_check_ans_bool
2394        {
2395          \int_gincr:N \g__enumext_count_item_anskey_int
2396        }
2397    }
```

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

10.26.4 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.

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.

```
2398  \cs_new_protected:Npn \__enumext_keyans_show_left:n #1
2399    {
2400      \tl_if_novalue:nF { #1 }
2401        {
```

```
2402            \tl_set:Ne \l__enumext_keyans_item_opt_tl { #1 }
2403          }
2404        \bool_if:NT \l__enumext_show_answer_bool
2405          {
2406            \__enumext_keyans_show_ans:
2407          }
2408        \bool_if:NT \l__enumext_show_position_bool
2409          {
2410            \__enumext_keyans_show_pos:
2411          }
2412      }
2413  \cs_new_protected:Nn \__enumext_keyans_show_item_opt:
2414      {
2415        \tl_if_empty:NF \l__enumext_keyans_item_opt_tl
2416          {
2417            \bool_lazy_or:nnT
2418              { \bool_if_p:N \l__enumext_show_answer_bool }
2419              { \bool_if_p:N \l__enumext_show_position_bool }
2420              {
2421                \__enumext_keyans_wrapper_opt:n { \l__enumext_keyans_item_opt_tl } \c_space_tl
2422              }
2423          }
2424      }
2425  \cs_new_protected:Nn \__enumext_keyans_show_ans:
2426      {
2427        \tl_put_left:Nn \l__enumext_label_v_tl
2428          {
2429            \__enumext_print_keyans_box:NN
2430              \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
2431          }
2432      }
2433  \cs_new_protected:Nn \__enumext_keyans_show_pos:
2434      {
2435        \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
2436          {
2437            \tl_set:Ne \l__enumext_mark_answer_sym_tl
2438              {
2439                \group_begin:
2440                \exp_not:N \normalfont
2441                \exp_not:N \footnotesize [ \int_eval:n
2442                  {
2443                    \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
2444                  }
2445                  ]
2446                \group_end:
2447              }
2448          }
2449          {
2450            \tl_set:Ne \l__enumext_mark_answer_sym_tl
2451              {
2452                \group_begin:
2453                \exp_not:N \normalfont
2454                \exp_not:N \footnotesize [ \int_eval:n
2455                  {
2456                    \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } + 1
2457                  }
2458                  ]
2459                \group_end:
2460              }
2461          }
2462        \tl_put_left:Nn \l__enumext_label_v_tl
2463          {
2464            \__enumext_print_keyans_box:NN
2465              \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
2466          }
2467      }
```

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

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

In order to have a cleaner implementation of `\item*` 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*`.

```
2468 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
2469   {
2470     \keys_define:nn { enumext / #1 }
2471       {
2472         item-sym* .tl_set:c  = { l__enumext_item_symbol_#2_tl },
2473         item-sym* .value_required:n = true,
2474         item-sym* .initial:n  = {$\star$},
2475         item-pos* .dim_set:c  = { l__enumext_item_symbol_sep_#2_dim },
2476         item-pos* .value_required:n = true,
2477       }
2478   }
2479 \clist_map_inline:nn
2480   {
2481     {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {enumext*}{vii}
2482   }
2483   { \__enumext_tmp:nn #1 }
```

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

10.28 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 with the `mini-env` key and 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.

```
2484 \cs_new_protected:Nn \__enumext_footnotetext:nn
2485   {
2486     \footnotetext[#1]{#2}
2487   }
2488 \cs_new_protected:Nn \__enumext_renew_footnote:
2489   {
2490     \seq_gclear:N \g__enumext_footnote_arg_seq
2491     \seq_gclear:N \g__enumext_footnote_int_seq
2492     \RenewDocumentCommand \footnote { o +m }
2493       {
2494         \tl_if_novalue:nTF {##1}
2495           {
2496             \stepcounter{footnote}
2497             \int_gset_eq:Nc \g__enumext_footnote_int { c@footnote }
2498           }
2499           {
2500             \int_gset:Nn \g__enumext_footnote_int { ##1 }
2501           }
2502         \footnotemark [ \g__enumext_footnote_int ]
2503         \seq_gput_right:Nn \g__enumext_footnote_arg_seq { ##2 }
2504         \seq_gput_right:NV \g__enumext_footnote_int_seq \g__enumext_footnote_int
2505       }
2506   }
2507 \cs_new_protected:Nn \__enumext_print_footnote:
2508   {
2509     \seq_if_empty:NF \g__enumext_footnote_int_seq
2510       {
2511         \seq_map_pairwise_function:NNN
2512           \g__enumext_footnote_int_seq
2513           \g__enumext_footnote_arg_seq
2514           \__enumext_footnotetext:nn
2515       }
2516   }
```

(*End of definition for* `\__enumext_footnotetext:nn`, `\__enumext_renew_footnote:`, *and* `\__enumext_print_footnote:`.)

10.29 Redefining `\item` command

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.

10.29.1   The \item command in enumext

__enumext_default_item:n

The \item and \item[⟨*custom*⟩] commands work in the usual way on enumext.

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_ans_bool set by the key check-ans, set the boolean variable \l__enumext_wrap_label_X_bool to "true" and execute __enumext_item_std:w.

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.

The boolean variable \l__enumext_wrap_label_X_bool is used by the function __enumext_make_label: (§10.30).

```
2517  \cs_new_protected:Npn \__enumext_default_item:n #1
2518    {
2519      \tl_if_novalue:nTF {#1}
2520        {
2521          \bool_if:NT \l__enumext_check_ans_bool
2522            {
2523              \int_gincr:N \g__enumext_count_item_number_int
2524            }
2525          \bool_set_true:c { l__enumext_wrap_label_ \__enumext_level: _bool }
2526          \__enumext_item_std:w \tl_use:c { l__enumext_fake_item_indent_ \__enumext_level: _tl }
2527        }
2528        {
2529          \bool_set_eq:cc
2530            { l__enumext_wrap_label_ \__enumext_level: _bool }
2531            { l__enumext_wrap_label_opt_ \__enumext_level: _bool }
2532          \__enumext_item_std:w [#1] \tl_use:c { l__enumext_fake_item_indent_ \__enumext_level: _tl
2533        }
2534    }
```

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

__enumext_starred_item:nn

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 set by the labelsep key and can be *offset* using 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 optional argument in the global variable \g__enumext_item_symbol_tl, followed by setting the variable \l__enumext_item_symbol_sep_X_dim set by the key item*-sep or by the second optional argument.

Then we will see the state of the variable \l__enumext_check_ans_bool set by the key check-ans, set the boolean variable \l__enumext_wrap_label_X_bool to "true" and execute __enumext_item_std:w.

In this function the optional argument of __enumext_item_std:w is omitted, we only want it to be numbered.

The boolean variable \l__enumext_wrap_label_X_bool and the vars \l__enumext_item_symbol_sep_X_dim, \g__enumext_item_symbol_tl are used by the function __enumext_make_label: (§10.30).

```
2535  \cs_new_protected:Npn \__enumext_starred_item:nn #1 #2
2536    {
2537      \tl_if_novalue:nF {#1}
2538        {
2539          \tl_set:cn { l__enumext_item_symbol_ \__enumext_level: _tl } {#1}
2540        }
2541      \tl_gset_eq:Nc \g__enumext_item_symbol_tl { l__enumext_item_symbol_ \__enumext_level: _tl }
2542      \tl_if_novalue:nTF {#2}
2543        {
2544          \dim_set_eq:cc
2545            { l__enumext_item_symbol_sep_ \__enumext_level: _dim }
2546            { l__enumext_labelsep_ \__enumext_level: _dim }
2547        }
2548        {
2549          \dim_set:cn { l__enumext_item_symbol_sep_ \__enumext_level: _dim } {#2}
2550        }
2551      \bool_if:NT \l__enumext_check_ans_bool
2552        {
2553          \int_gincr:N \g__enumext_count_item_number_int
```

```
2554          }
2555      \bool_set_true:c { l__enumext_wrap_label_ \__enumext_level: _bool }
2556      \__enumext_item_std:w \tl_use:c { l__enumext_fake_item_indent_ \__enumext_level: _tl }
2557    }
```

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

__enumext_redefine_item: The function __enumext_redefine_item: will redefine the \item command in the enumext environment for the internal mechanism of check-answers for check-ans key and adding the starred \item* version.

This function is passed to __enumext_list_arg_two_X: which is used in the definition of the enumext environment (§10.31.2).

```
2558 \cs_new_protected:Nn \__enumext_redefine_item:
2559    {
2560      \RenewDocumentCommand \item { s o o }
2561        {
2562          \bool_if:nTF {##1}
2563            {
2564              \__enumext_starred_item:nn {##2} {##3}
2565            }
2566            { \__enumext_default_item:n {##2} }
2567        }
2568    }
```

(*End of definition for* __enumext_redefine_item:.)

10.29.2 The \item command in keyans

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.

```
2569 \cs_new_protected:Npn \__enumext_keyans_default_item:n #1
2570    {
2571      \tl_if_novalue:nTF { #1 }
2572        {
2573          \bool_set_true:N \l__enumext_wrap_label_v_bool
2574          \__enumext_item_std:w \tl_use:N \l__enumext_fake_item_indent_v_tl
2575        }
2576        {
2577          \bool_set_eq:NN \l__enumext_wrap_label_v_bool \l__enumext_wrap_label_opt_v_bool
2578          \__enumext_item_std:w [#1] \tl_use:N \l__enumext_fake_item_indent_v_tl
2579        }
2580    }
```

(*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⟩.

```
2581 \cs_new_protected:Npn \__enumext_keyans_starred_item:n #1
2582    {
2583      \tl_set_eq:NN \l__enumext_keyans_tmpa_tl \l__enumext_label_v_tl
2584      \__enumext_keyans_show_left:n { #1 }
2585      \bool_set_true:N \l__enumext_wrap_label_v_bool
2586      \__enumext_item_std:w \tl_use:N \l__enumext_fake_item_indent_v_tl \__enumext_keyans_show_item
```

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 and finally *store* it in the ⟨sequence⟩.

```
2587      \tl_set_eq:NN \l__enumext_label_v_tl \l__enumext_keyans_tmpa_tl
2588      \__enumext_keyans_addto_prop:n { #1 }
2589      \__enumext_keyans_store_ref:
2590      \__enumext_keyans_addto_seq:n { #1 }
2591      \int_gincr:N \g__enumext_check_starred_cmd_int
2592    }
```

(*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* 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 is passed to __enumext_list_arg_two_v: which is used in the definition of the keyans environment (§10.31.2).

```
2593 \cs_new_protected:Nn \__enumext_keyans_redefine_item:
2594   {
2595     \RenewDocumentCommand \item { s o }
2596       {
2597         \bool_if:nTF {##1}
2598           {
2599             \peek_remove_spaces:n
2600               {
2601                 \__enumext_keyans_starred_item:n {##2}
2602               }
2603           }
2604           {
2605             \__enumext_keyans_default_item:n {##2}
2606           }
2607       }
2608   }
```

(*End of definition for* \item* *and* __enumext_keyans_redefine_item:. *This function is documented on page 11*.)

10.30 Redefining \makelabel command

Redefine \makelabel for the keys align, font, wrap-label, wrap-label* and \item* for enumext and keyans environments.

10.30.1 Redefining \makelabel for enumext

__enumext_item_starred:

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

```
2609 \cs_new_protected:Nn \__enumext_item_starred:
2610   {
2611     \tl_if_empty:cF { l__enumext_item_symbol_ \__enumext_level: _tl }
2612       {
2613         \mode_leave_vertical:
2614         \skip_horizontal:n { -\dim_use:c { l__enumext_item_symbol_sep_ \__enumext_level: _dim } }
2615         \makebox[ 0pt ][ r ]{ \g__enumext_item_symbol_tl }
2616         \skip_horizontal:n { \dim_use:c { l__enumext_item_symbol_sep_ \__enumext_level: _dim } }
2617       }
2618   }
```

(*End of definition for* __enumext_item_starred:.)

__enumext_make_label:

The function __enumext_make_label: redefine \makelabel for the enumext environment.

This function is passed to __enumext_list_arg_two_X: which is used in the definition of the enumext environment (§10.31.2).

```
2619 \cs_new_protected:Nn \__enumext_make_label:
2620   {
2621     \RenewDocumentCommand \makelabel { m }
2622       {
2623         \tl_use:c { l__enumext_label_fill_left_ \__enumext_level: _tl }
2624         \tl_use:c { l__enumext_label_font_style_ \__enumext_level: _tl }
2625         \bool_if:cTF { l__enumext_wrap_label_ \__enumext_level: _bool }
2626           {
2627             \__enumext_item_starred:
2628             \use:c { __enumext_wrapper_label_ \__enumext_level: :n } { ##1 }
2629           }
2630           { ##1 }
2631         \tl_use:c { l__enumext_label_fill_right_ \__enumext_level: _tl }
2632         \tl_gclear:N \g__enumext_item_symbol_tl
2633       }
2634   }
```

(*End of definition for* __enumext_make_label:.)

10.30.2 Redefining \makelabel for keyans

__enumext_keyans_make_label:

The function __enumext_keyans_make_label: redefine \makelabel for keyans environment.

This function is passed to __enumext_list_arg_two_v: which is used in the definition of the keyans environment (§10.31.2).

```
2635 \cs_new_protected:Nn \__enumext_keyans_make_label:
2636   {
2637     \RenewDocumentCommand \makelabel { m }
2638       {
2639         \tl_use:N \l__enumext_label_fill_left_v_tl
2640         \tl_use:N \l__enumext_label_font_style_v_tl
2641         \bool_if:NTF \l__enumext_wrap_label_v_bool
2642           {
2643             \__enumext_wrapper_label_v:n { ##1 }
2644           }
2645           { ##1 }
2646         \tl_use:N \l__enumext_label_fill_right_v_tl
2647       }
2648   }
```

(*End of definition for* __enumext_keyans_make_label:.)

10.31 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.

10.31.1 Calculation of \leftmargin and \itemindent

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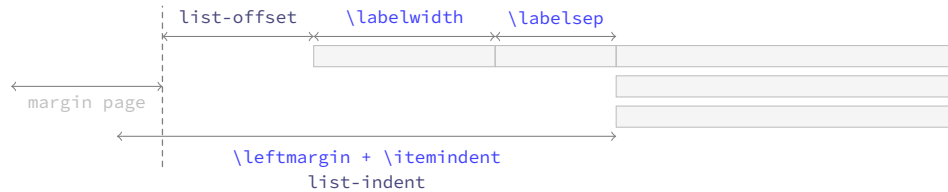
Figure 9: Representation of standard horizontal lengths in list environment.

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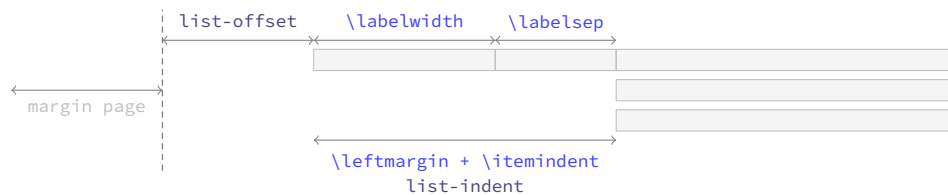
Figure 10: Representation of horizontal lengths concept in list in enumext.

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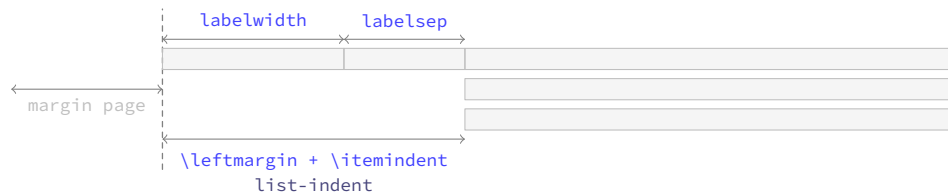
Figure 11: Default horizontal lengths in enumext.

__enumext_calc_hspace:NNNNNNN
__enumext_calc_hspace:ccccccc

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

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

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

This function is passed to `\__enumext_list_arg_two_X:` which is used in the definition of the `enumext` and `keyans` environments (§10.31.2).

```
2649  \cs_new_protected:Npn \__enumext_calc_hspace:NNNNNNN #1 #2 #3 #4 #5 #6 #7
2650    {
2651      \dim_compare:nNnT { #1 } < { \c_zero_dim }
2652        {
2653          \msg_warning:nnnV { enumext } { width-non-positive }{ labelwidth }{ #1 }
2654          \dim_set:Nn #1 { \dim_abs:n { #1 } }
2655        }
2656      \dim_compare:nNnT { #2 } < { \c_zero_dim }
2657        {
2658          \msg_warning:nnnV { enumext } { width-negative }{ labelsep }{ #2 }
2659          \dim_set:Nn #2 { \dim_abs:n { #2 } }
2660        }
```

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

```
2661      \bool_if:nF #7 { \dim_set:Nn #4 { #1 + #2} }
```

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

```
2662      \dim_compare:nNnTF { #4 } < { \c_zero_dim }
2663        {
2664          \dim_set:Nn #6 { #1 + #2 - #4}
2665          \dim_set:Nn #5 { #1 + #2 + #3 - #6 }
2666        }
2667        {
2668          \dim_compare:nNnT { #4 } = { #1 + #2 }
2669            { \dim_set:Nn #6 { \c_zero_dim } }
2670          \dim_compare:nNnT { #4 } < { #1 + #2 }
2671            { \dim_set:Nn #6 { #1 + #2 - #4} }
2672          \dim_compare:nNnT { #4 } > { #1 + #2 }
2673            {
2674              \dim_set:Nn #6 { -#1 - #2 + #4}
2675              \dim_set:Nn #6 { #6*-1}
2676            }
2677          \dim_set:Nn #5 { #1 + #2 + #3 - #6 }
2678        }
2679    }
2680  \cs_generate_variant:Nn \__enumext_calc_hspace:NNNNNNN { ccccccc }
```

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

10.31.2 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.

```
2681  \cs_set_protected:Npn \__enumext_tmp:n #1
2682    {
2683      \cs_new_protected:cpn { __enumext_list_arg_two_#1: }
2684        {
2685          \__enumext_calc_hspace:ccccccc
2686            { l__enumext_labelwidth_#1_dim } { l__enumext_labelsep_#1_dim }
2687            { l__enumext_listoffset_#1_dim } { l__enumext_leftmargin_tmp_#1_dim }
2688            { l__enumext_leftmargin_#1_dim } { l__enumext_itemindent_#1_dim }
2689            { l__enumext_leftmargin_tmp_#1_bool }
2690          \clist_map_inline:nn
2691            { labelsep, labelwidth, itemindent, leftmargin, rightmargin, listparindent }
2692            { \dim_set_eq:cc {####1} { l__enumext_####1_#1_dim } }
2693          \clist_map_inline:nn { topsep, parsep, partopsep, itemsep }
2694            { \skip_set_eq:cc {####1} { l__enumext_####1_#1_skip } }
2695          \usecounter { enumX#1 }
2696          \setcounter { enumX#1 } { \int_eval:n { \int_use:c { l__enumext_start_#1_int } - 1 } }
2697          \str_if_eq:nnTF {#1} { v }
2698            {
2699              \__enumext_keyans_redefine_item:
2700              \__enumext_keyans_make_label:
2701              \__enumext_keyans_ref:
2702              \__enumext_keyans_fake_item:
2703              \bool_if:cT { l__enumext_show_length_#1_bool }
2704                {
```

```
2705                    \msg_term:nnnn { enumext } { list-lengths-not-nested } { v } { keyans }
2706                }
2707              }
2708              {
2709                \__enumext_redefine_item:
2710                \__enumext_make_label:
2711                \__enumext_standar_ref:
2712                \__enumext_fake_item:
2713                \bool_if:cT { l__enumext_show_length_#1_bool }
2714                  {
2715                    \msg_term:nnne { enumext } { list-lengths } {#1} { \int_use:N \l__enumext_level_in
2716                  }
2717              }
2718          }
2719      }
2720  \clist_map_inline:nn { i, ii, iii, iv, v } { \__enumext_tmp:n {#1} }
```

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

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

```
2721  \cs_set_protected:Npn \__enumext_tmp:n #1
2722    {
2723      \cs_new_protected:cpn { __enumext_list_arg_two_#1: }
2724        {
2725          \__enumext_calc_hspace:ccccccc
2726            { l__enumext_labelwidth_#1_dim } { l__enumext_labelsep_#1_dim }
2727            { l__enumext_listoffset_#1_dim } { l__enumext_leftmargin_tmp_#1_dim }
2728            { l__enumext_leftmargin_#1_dim } { l__enumext_itemindent_#1_dim }
2729            { l__enumext_leftmargin_tmp_#1_bool }
2730          \clist_map_inline:nn
2731            { labelsep, labelwidth, itemindent, leftmargin, rightmargin, listparindent }
2732            { \dim_set_eq:cc {####1} { l__enumext_####1_#1_dim } }
2733          \clist_map_inline:nn { topsep, parsep, partopsep, itemsep }
2734            { \skip_set_eq:cc {####1} { l__enumext_####1_#1_skip } }
2735          \skip_set_eq:Nc \parsep  { l__enumext_itemsep_#1_skip }
2736          \skip_zero:N \partopsep
2737          \usecounter { enumX#1 }
2738          \setcounter { enumX#1 } { \int_eval:n { \int_use:c { l__enumext_start_#1_int } - 1 } }
2739          \__enumext_starred_ref:
2740          \str_if_eq:nnTF {#1} { vii }
2741            {
2742              \__enumext_fake_item_vii:
2743              \bool_if:cT { l__enumext_show_length_vii_bool }
2744                { \msg_term:nnnn { enumext } { list-lengths-not-nested } { vii } { enumext* } }
2745            }
2746            {
2747              \__enumext_fake_item_viii:
2748              \bool_if:cT { l__enumext_show_length_#1_bool }
2749                { \msg_term:nnnn { enumext } { list-lengths-not-nested } { #1 } { keyans* } }
2750            }
2751        }
2752    }
2753  \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:.)

10.32   The environment enumext

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

```
2754  \NewDocumentEnvironment{enumext}{ O{} }
2755    {
2756      \__enumext_safe_exec:
2757      \__enumext_parse_keys:n {#1}
2758      \__enumext_before_list:
2759      \__enumext_start_store_level:
2760      \__enumext_start_list:nn
2761        { \tl_use:c { l__enumext_label_ \__enumext_level: _tl } }
2762        {
2763          \use:c { __enumext_list_arg_two_ \__enumext_level: : }
```

```
2764            \__enumext_before_keys_exec:
2765          }
2766        \__enumext_after_args_exec:
2767      }
2768      {
2769        \__enumext_stop_list:
2770        \__enumext_stop_store_level:
2771        \__enumext_after_list:
2772      }
```

(*End of definition for enumext. This function is documented on page 4.*)

__enumext_safe_exec: The __enumext_safe_exec: function first execute the function __enumext_is_not_nested: which
will set the variable \g__enumext_standard_bool to *"true"* if the environment is not nested in enumext*,
we increment the variable \l__enumext_level_int for the nesting levels and set the \l__enumext_-
standard_bool variable to *"true"*. Finally we set the variable \l__enumext_standar_first_level_-
bool to *"true"* only if the environment is not nested and we are at the *"first level"* of it using the function
__enumext_is_on_first_level:.

```
2773  \cs_new_protected:Nn \__enumext_safe_exec:
2774    {
2775      \__enumext_is_not_nested:
2776      \int_incr:N \l__enumext_level_int
2777      \int_compare:nNnT { \l__enumext_level_int } > { 4 }
2778        { \msg_fatal:nn { enumext } { list-too-deep } }
2779      \bool_set_true:N \l__enumext_standar_bool
2780      \__enumext_is_on_first_level:
2781    }
```

(*End of definition for __enumext_safe_exec:.*)

__enumext_parse_keys:n   The __enumext_parse_store_keys:n function will parse the ⟨keys⟩ passed to the optional environ-
ment argument enumext by levels only if present. First we clear the variable \l__enumext_series_str
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, otherwise we will pass the ⟨keys⟩ to the inner
levels of the environment and finally if the variable \l__enumext_store_active_bool established by
the key save-ans is true we execute __enumext_parse_store_keys:n used by the key save-key.

```
2782  \cs_new_protected:Npn \__enumext_parse_keys:n #1
2783    {
2784      \tl_if_novalue:nF {#1}
2785        {
2786          \str_clear:N \l__enumext_series_str
2787          \int_compare:nNnTF { \l__enumext_level_int } = { 1 }
2788            {
2789              \keys_set:nn { enumext / level-1 } {#1}
2790              \__enumext_parse_series:n {#1}
2791            }
2792            {
2793              \exp_args:Ne \keys_set:nn
2794                { enumext / level-\int_use:N \l__enumext_level_int } {#1}
2795            }
2796          \bool_if:NT \l__enumext_store_active_bool
2797            {
2798              \__enumext_parse_store_keys:n {#1}
2799            }
2800        }
2801    }
```

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

__enumext_parse_store_keys:n   The function __enumext_parse_store_keys:n searches for the values of the columns and columns-
sep keys in the optional arguments per-level in enumext environment as long as the starred versions
of the columns* and columns-sep* keys are not active. The captured values are stored in the variable
\l__enumext_store_opt_X_tl which is used by the function __enumext_store_level_open:.

```
2802  \cs_new_protected:Npn \__enumext_parse_store_keys:n #1
2803    {
2804      \bool_if:cF { l__enumext_store_columns_ \__enumext_level: _bool }
2805        {
2806          \regex_match:nnT { \b columns\b } {#1}
2807            {
2808              \int_set_eq:cc
```

```
2809                { l__enumext_store_columns_ \__enumext_level: _int }
2810                { l__enumext_columns_  \__enumext_level: _int }
2811              \tl_put_right:ce { l__enumext_store_opt_ \__enumext_level: _tl }
2812                {
2813                  columns = \exp_not:v { l__enumext_store_columns_ \__enumext_level: _int },
2814                }
2815            }
2816          }
2817        \bool_if:cF { l__enumext_store_columns_sep_ \__enumext_level: _bool }
2818          {
2819            \regex_match:nnT { \b columns-sep \b} {#1}
2820              {
2821                \dim_set_eq:cc
2822                  { l__enumext_store_columns_sep_ \__enumext_level: _dim }
2823                  { l__enumext_columns_sep_  \__enumext_level: _dim }
2824                \tl_put_right:ce { l__enumext_store_opt_ \__enumext_level: _tl }
2825                  {
2826                    columns-sep = \exp_not:v { l__enumext_store_columns_sep_ \__enumext_level: _dim }
2827                  }
2828              }
2829          }
2830      }
```

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

__enumext_start_store_level: The __enumext_start_store_level: and __enumext_stop_store_level: functions activate the
__enumext_stop_store_level: level saving mechanism for storage in ⟨*sequence*⟩ of the \anskey command.

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

```
2831  \cs_new_protected:Nn \__enumext_start_store_level:
2832    {
2833      \bool_lazy_all:nT
2834        {
2835          { \bool_if_p:N \l__enumext_store_active_bool }
2836          { \bool_not_p:n { \l__enumext_keyans_env_bool } }
2837          { \bool_not_p:n { \g__enumext_starred_bool } }
2838        }
2839        {
2840          \int_compare:nNnT { \l__enumext_level_int } > { 1 }
2841            {
2842              \bool_set_true:c { l__enumext_store_upper_level_ \__enumext_level: _bool }
2843              \__enumext_store_level_open:
2844            }
2845        }
2846      \bool_lazy_all:nT
2847        {
2848          { \bool_if_p:N \l__enumext_store_active_bool }
2849          { \bool_not_p:n { \l__enumext_keyans_env_bool } }
2850          { \bool_if_p:N \g__enumext_starred_bool }
2851        }
2852        {
2853          \int_compare:nNnT { \l__enumext_level_int } > { 0 }
2854            {
2855              \bool_set_true:c { l__enumext_store_upper_level_ \__enumext_level: _bool }
2856              \__enumext_store_level_open:
2857            }
2858        }
2859    }
2860  \cs_new_protected:Nn \__enumext_stop_store_level:
2861    {
2862      \bool_if:cT { l__enumext_store_upper_level_ \__enumext_level: _bool }
2863        {
2864          \__enumext_store_level_close:
2865        }
2866    }
```

(*End of definition for* __enumext_start_store_level: *and* __enumext_stop_store_level:.)

__enumext_before_list: The function __enumext_before_list: 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.

```
2867  \cs_new_protected:Nn \__enumext_before_list:
```

```
2868    {
2869        \__enumext_vspace_above:
2870        \__enumext_before_args_exec:
```

The function `\__enumext_check_ans:` will handle the check answer mechanism, which will be activated with the check-ans key.

```
2871        \__enumext_check_ans:
```

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_env*` 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_env*` environment on the *"left side"*, always having a current `\linewidth` as *maximum width* between them.

```
2872        \dim_compare:nNnT
2873        { \dim_use:c { l__enumext_minipage_right_ \__enumext_level: _dim } } > { \c_zero_dim }
2874        {
2875            \dim_set:cn { l__enumext_minipage_left_ \__enumext_level: _dim }
2876            {
2877                \linewidth
2878                - \dim_use:c { l__enumext_minipage_right_ \__enumext_level: _dim }
2879                - \dim_use:c { l__enumext_minipage_hsep_ \__enumext_level: _dim }
2880            }
```

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_mini_addvspace:` is called and the `__enumext_mini_env*` 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.

💣 Here we use the plain TEX macro `\nointerlineskip` to prevent baseline *"glue"* being added between the next pair of boxes in a *vertical list*.

```
2881            \bool_set_true:c { l__enumext_minipage_active_ \__enumext_level: _bool }
2882            \int_gincr:N \g__enumext_minipage_stat_int
2883            \__enumext_mini_addvspace:
2884            \nointerlineskip\noindent
2885            \begin{__enumext_mini_env*}
2886            { \dim_use:c { l__enumext_minipage_left_ \__enumext_level: _dim } }
2887        }
2888        \__enumext_multicols_start:
2889    }
```

(End of definition for `\__enumext_before_list:`.)

`\__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.

```
2890    \cs_new_protected:Nn \__enumext_multicols_start:
2891    {
2892        \int_compare:nNnT
2893        { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } > { 1 }
2894        {
2895            \dim_compare:nNnT
2896            { \dim_use:c { l__enumext_columns_sep_ \__enumext_level: _dim } } = { \c_zero_dim }
2897            {
2898                \dim_set:cn { l__enumext_columns_sep_ \__enumext_level: _dim }
2899                {
2900                    ( \dim_use:c { l__enumext_labelwidth_ \__enumext_level: _dim }
2901                    + \dim_use:c { l__enumext_labelsep_ \__enumext_level: _dim }
2902                    ) / \int_use:c { l__enumext_columns_ \__enumext_level: _int }
2903                    - \dim_use:c { l__enumext_listoffset_ \__enumext_level: _dim }
2904                }
2905            }
2906            \dim_set_eq:Nc \columnsep { l__enumext_columns_sep_ \__enumext_level: _dim }
2907            \skip_zero:N \multicolsep
2908            \int_compare:nNnT { \l__enumext_level_int } > { 1 }
2909            {
2910                \dim_zero:N \columnseprule
2911            }
```

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.

```
2912          \bool_if:cF { l__enumext_minipage_active_ \__enumext_level: _bool }
2913            {
2914              \__enumext_multi_addvspace:
2915            }
2916          \raggedcolumns
2917          \begin{multicols}{ \int_use:c { l__enumext_columns_ \__enumext_level: _int } }
2918        }
2919    }
```

(*End of definition for* __enumext_multicols_start:.)

`\__enumext_multicols_stop:` The function `\__enumext_multicols_stop:` will stop the `multicols` environment. If the boolean variable `\l__enumext_minipage_active_X_bool` is false (not nested in `__enumext_mini_env*`) we will apply our *"vertical adjust"* spacing.

```
2920  \cs_new_protected:Nn \__enumext_multicols_stop:
2921    {
2922      \int_compare:nNnT
2923        { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } > { 1 }
2924        {
2925          \end{multicols}
2926          \bool_if:cF { l__enumext_minipage_active_ \__enumext_level: _bool }
2927            {
2928              \par\addvspace{ \skip_use:c { l__enumext_multicols_below_ \__enumext_level: _skip } }
2929            }
2930        }
2931    }
```

(*End of definition for* __enumext_multicols_stop:.)

`\__enumext_after_list:` The function `\__enumext_after_list:` will will 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_env*` environment has not been closed), then close `__enumext_mini_env*` and add the *adjusted vertical space* `\l__enumext_minipage_after_skip`, otherwise we will close the `multicols` environment.

```
2932  \cs_new_protected:Nn \__enumext_after_list:
2933    {
2934      \bool_if:cTF { l__enumext_minipage_active_ \__enumext_level: _bool }
2935        {
2936          \int_compare:nNnT { \g__enumext_minipage_stat_int } = { 1 }
2937            {
2938              \msg_warning:nn { enumext } { missing-miniright }
2939              \miniright
2940            }
2941          \int_gzero:N \g__enumext_minipage_stat_int
2942          \end{__enumext_mini_env*}
2943          \par\addvspace { \l__enumext_minipage_after_skip }
2944        }
2945        { \__enumext_multicols_stop: }
```

If the `check-ans` key is active, we set the boolean variable `\g__enumext_check_ans_show_bool` to true and copy the *"store name"* to the variable `\g__enumext_store_name_tl`.

```
2946        \__enumext_check_ans_to_hook:
```

Now apply the {⟨*code*⟩} handled by the `after` key together with the *vertical space* handled by the `below` key if they are present, set `\l__enumext_standar_bool` to false and save the *current value* of the counter for `series`, `resume` and `resume*` keys.

```
2947        \__enumext_after_stop_list:
2948        \__enumext_vspace_below:
2949        \bool_set_false:N \l__enumext_standar_bool
2950        \__enumext_resume_save_counter:
2951    }
```

(*End of definition for* __enumext_after_list:.)

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`.

```
2952  \__enumext_after_env:nn {enumext} { \__enumext_check_ans_show: }
```

10.33   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"*.

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

```
2953  \NewDocumentEnvironment{keyans}{ O{} }
2954    {
2955      \__enumext_keyans_safe_exec:
2956      \__enumext_keyans_parse_keys:n {#1}
2957      \__enumext_before_list_v:
2958      \__enumext_start_list:nn
2959        { \tl_use:N \l__enumext_label_v_tl }
2960        {
2961          \__enumext_list_arg_two_v:
2962          \__enumext_before_keys_exec_v:
2963        }
2964      \__enumext_after_args_exec_v:
2965    }
2966    {
2967      \__enumext_check_starred_cmd:n { item }
2968      \__enumext_stop_list:
2969      \__enumext_after_list_v:
2970    }
```

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

__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.

```
2971  \cs_new_protected:Nn \__enumext_keyans_safe_exec:
2972    {
2973      \bool_if:NF \l__enumext_store_active_bool
2974        {
2975          \msg_error:nnnn { enumext } { wrong-place }{ keyans }{ save-ans }
2976        }
2977      \int_incr:N \l__enumext_keyans_level_int
2978      \bool_set_true:N \l__enumext_keyans_env_bool
2979      \__enumext_keyans_save_start_line:
2980      % Set false for interfering with enumext nested in keyans (yes, its possible and crayze)
2981      \bool_set_false:N \l__enumext_store_active_bool
2982      \int_compare:nNnT { \l__enumext_keyans_level_int } > { 1 }
2983        {
2984          \msg_error:nn { enumext } { keyans-nested }
2985        }
2986      \int_compare:nNnT { \l__enumext_level_int } > { 1 }
2987        {
2988          \msg_error:nn { enumext } { keyans-wrong-level }
2989        }
2990    }
```

(*End of definition for* __enumext_keyans_safe_exec:.)

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

```
2991  \cs_new_protected:Npn \__enumext_keyans_parse_keys:n #1
2992    {
2993      \keys_set:nn { enumext / keyans } {#1}
2994    }
```

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

__enumext_before_list_v: The function __enumext_before_list_v: will add the *vertical spacing above* the environment if the above key is active next to the ⟨*code*⟩ defined by the before key if it is active.

```
2995  \cs_new_protected:Nn \__enumext_before_list_v:
2996    {
2997      \__enumext_vspace_above_v:
2998      \__enumext_before_args_exec_v:
```

When the `mini-env` key is active it will set the value of the `\l__enumext_minipage_right_v_dim` to be the *width* of the `__enumext_mini_env*` environment on the *left side*, using this value together with the value of the `\l__enumext_minipage_hsep_v_dim` set by the `mini-sep` key, the value of `\l__enumext_minipage_left_v_dim` will be set, which will be the *width* of `__enumextt_mini_env*` environment on the *right side*, always having `\linewidth` as the maximum width between them.

```
2999    \dim_compare:nNnT { \l__enumext_minipage_right_v_dim } > { \c_zero_dim }
3000        {
3001            \dim_set:Nn \l__enumext_minipage_left_v_dim
3002                {
3003                    \linewidth - \l__enumext_minipage_right_v_dim - \l__enumext_minipage_hsep_v_dim
3004                }
```

The boolean variable `\l__enumext_minipage_active_v_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_keyans_mini_addvspace:` is called and the `__enumext_mini_env*` environment on *left side* will be initialized followed by the *vertical spacing* `\l__enumext_minipage_left_skip`. Here we use the plain TeX macro `\nointerlineskip` to prevent baseline *"glue"* being added between the next pair of boxes in a *vertical list*.

```
3005            \bool_set_true:N \l__enumext_minipage_active_v_bool
3006            \int_gincr:N \g__enumext_minipage_stat_int
3007            \__enumext_keyans_mini_addvspace:
3008            \nointerlineskip\noindent
3009            \begin{__enumext_mini_env*}{ \l__enumext_minipage_left_v_dim }
3010        }
```

After these actions, the `\__enumext_keyans_multicols_start:` function is called to handle the `multicols` environment.

```
3011    \__enumext_keyans_multicols_start:
3012    }
```

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

`\__enumext_keyans_multicols_start:` The function `\__enumext_keyans_multicols_start:` will start the `multicols` environment according to the value passed by the `columns` key.

```
3013 \cs_new_protected:Nn \__enumext_keyans_multicols_start:
3014    {
3015        \int_compare:nNnT { \l__enumext_columns_v_int } > { 1 }
3016            {
```

Set the default value for `\columnsep` when `columns-sep` key is `0pt`.

```
3017            \dim_compare:nNnT { \l__enumext_columns_sep_v_dim } = { \c_zero_dim }
3018                {
3019                    \dim_set:Nn \l__enumext_columns_sep_v_dim
3020                        {
3021                            (
3022                                \l__enumext_labelwidth_v_dim + \l__enumext_labelsep_v_dim
3023                            ) / \l__enumext_columns_v_int
3024                            - \l__enumext_listoffset_v_dim
3025                        }
3026                }
3027            \dim_set_eq:NN \columnsep \l__enumext_columns_sep_v_dim
```

Then we will set the value of `\multicolsep` and `\columnseprule` equal to zero (we do not want a vertical rule in this environment).

```
3028            \skip_zero:N \multicolsep
3029            \dim_zero:N \columnseprule
```

We will calculate the *vertical spacing* settings for the `multicols` environment using the function `\__enumext_keyans_multi_addvspace:` and apply our *"vertical adjust spacing"*, then start the `multicols` environment.

```
3030            \bool_if:NF \l__enumext_minipage_active_v_bool
3031                {
3032                    \__enumext_keyans_multi_addvspace:
3033                }
3034            \raggedcolumns
3035            \begin{multicols}{ \l__enumext_columns_v_int }
3036        }
3037    }
```

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

__enumext_keyans_multicols_stop:

The function `\__enumext_keyans_multicols_stop:` will stop the multicols environment. If the boolean variable `\l__enumext_minipage_active_v_bool` is false (not nested in `__enumext_mini_-env*`) we will apply our vertical "adjust" spacing.

```
3038 \cs_new_protected:Nn \__enumext_keyans_multicols_stop:
3039   {
3040     \int_compare:nNnT { \l__enumext_columns_v_int } > { 1 }
3041       {
3042         \end{multicols}
3043         \bool_if:NF \l__enumext_minipage_active_v_bool
3044           {
3045             \par\addvspace{ \l__enumext_multicols_below_v_skip }
3046           }
3047       }
3048   }
```

(*End of definition for* __enumext_keyans_multicols_stop:.)

__enumext_after_list_v:

The function `\__enumext_after_list_v:` will will check the state of the boolean variable `\l__-enumext_minipage_active_v_bool`, if it is "true" a small test will be executed to check if we have omitted the use of `\miniright` (the `__enumext_mini_env*` environment has not been closed), then close `__enumext_mini_env*` and add the vertical adjustment space `\l__enumext_minipage_after_-skip`, otherwise we will close the multicols environment.

```
3049 \cs_new_protected:Nn \__enumext_after_list_v:
3050   {
3051     \bool_if:NTF \l__enumext_minipage_active_v_bool
3052       {
3053         \int_compare:nNnT { \g__enumext_minipage_stat_int } = { 1 }
3054           {
3055             \msg_warning:nn { enumext } { missing-miniright }
3056             \miniright
3057           }
3058         \int_gzero:N \g__enumext_minipage_stat_int
3059         \end{__enumext_mini_env*}
3060         \par\addvspace{ \l__enumext_minipage_after_skip }
3061       }
3062     { \__enumext_keyans_multicols_stop: }
```

Finally we will apply the {⟨*code*⟩} handled by the after key together with the *vertical space* handled by the below key if they are present.

```
3063     \bool_set_false:N \l__enumext_keyans_env_bool
3064     \__enumext_after_stop_list_v:
3065     \__enumext_vspace_below_v:
3066   }
```

(*End of definition for* __enumext_after_list_v:.)

10.34 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 and are placed inside minipage environments, with the ⟨*label*⟩ underneath, adjusting widths according to the options passed to the environment.

Again it is necessary to "adjust" the spacing, both vertical and horizontal, to obtain an output like the one shown in the figure 12.
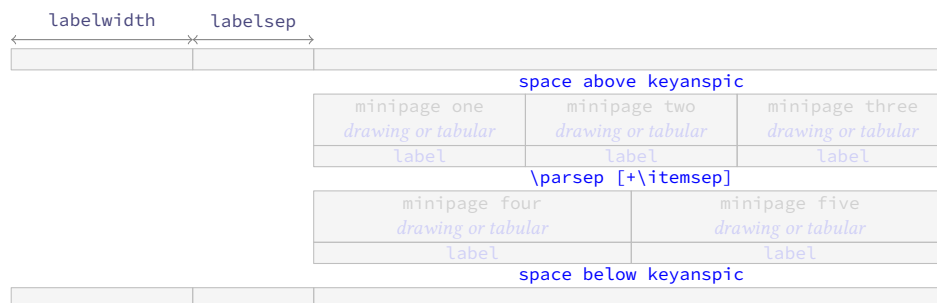
Figure 12: Representation of the keyanspic spacing in enumext.

This implementation is adapted from the answer given by Enrico Gregorio in How to process the body of an environment and divide it by a \macro?.

10.34.1 The command \anspic

\anspic

The \anspic command take three arguments, the starred (*) versions \anspic* and \anspic*[⟨content⟩] store the current ⟨label⟩ next to the [⟨content⟩] if it is present in the ⟨sequence⟩ and ⟨prop list⟩ defined by save-ans key. This command is used as a replacement for \item in the keyanspic environment.

```
3067 \NewDocumentCommand \anspic { s o +m }
3068   {
```

We check that the command is active in the keyanspic environment only if the save-ans key is present, otherwise we return an error.

```
3069     \bool_if:NF \l__enumext_store_active_bool
3070       {
3071         \msg_error:nnnn { enumext } { wrong-place }{ keyanspic }{ save-ans }
3072       }
3073     \int_compare:nNnT { \l__enumext_level_int } > { 1 }
3074       {
3075         \msg_error:nn { enumext } { keyanspic-wrong-level }
3076       }
3077     \int_compare:nNnT { \l__enumext_keyans_level_int } = { 1 }
3078       {
3079         \msg_error:nnnn { enumext } { command-wrong-place }{ anspic }{ keyans }
3080       }
```

The three arguments are handled by the function __enumext_keyans_anspic_code:nnn and stored in the sequence \l__enumext_keyans_pic_body_seq which is processed by the keyanspic environment.

```
3081     \seq_put_right:Nn \l__enumext_keyans_pic_body_seq
3082       {
3083         \__enumext_keyans_anspic_code:nnn { #1 } { #2 } { #3 }
3084       }
3085   }
```

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

__enumext_keyans_anspic_code:nnn

The function __enumext_keyans_anspic_code:nnn will be in charge of handling the *"counter"* and ⟨label⟩, which will have the same configuration as the keyans environment.

```
3086 \cs_new_protected:Nn \__enumext_keyans_anspic_code:nnn
3087   {
3088     \stepcounter { enumXvi }
3089     #3 \\
3090     \bool_if:nT { #1 }
3091       {
3092         \__enumext_keyans_addto_prop:n { #2 }
3093         \__enumext_keyans_store_ref:
3094         \__enumext_keyans_addto_seq:n { #2 }
3095         \int_gincr:N \g__enumext_check_starred_cmd_int
3096         \bool_lazy_or:nnT
3097           { \bool_if_p:N \l__enumext_show_answer_bool }
3098           { \bool_if_p:N \l__enumext_show_position_bool }
3099           {
3100             \tl_set_eq:NN \l__enumext_label_v_tl \l__enumext_label_vi_tl
3101             \__enumext_keyans_show_left:n { #2 }
3102             \tl_set_eq:NN \l__enumext_label_vi_tl \l__enumext_label_v_tl
3103           }
3104       }
3105     \tl_use:N \l__enumext_label_font_style_v_tl
3106     \__enumext_wrapper_label_v:n { \l__enumext_label_vi_tl } \__enumext_keyans_show_item_opt:
3107   }
```

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

10.34.2 The environment keyanspic

keyanspic

Now we define the environment keyanspic based on list. The optional argument [⟨*number above, number below*⟩] will determine the number of minipage environments that will be above and below separated by \parsep+\itemsep within it.

```
3108 \NewDocumentEnvironment{keyanspic}{ o }
3109   {
3110     \__enumext_keyans_pic_safe_exec:
3111     \__enumext_start_list:nn
3112       { }
3113       {
```

```
3114            \__enumext_keyans_pic_arg_two:
3115        }
```

We apply the "adjusted" vertical spacing above the environment

```
3116        \vspace { \l__enumext_keyans_pic_above_skip }
3117    }
```

If the optional argument is not present, the number of times the `\anspic` command appears will be counted from `\l__enumext_keyans_pic_body_seq` and placed in `minipage` environments on a single line. Finally we check if `\anspic*` has been used, set the counter to zero and apply our "adjusted" vertical space below the environment.

```
3118    {
3119        \tl_if_novalue:nTF { #1 }
3120          {
3121            \__enumext_keyans_pic_do:e { \seq_count:N \l__enumext_keyans_pic_body_seq }
3122          }
3123          { \__enumext_keyans_pic_do:n { #1 } }
3124        \__enumext_stop_list:
3125        \__enumext_check_starred_cmd:n { anspic }
3126        \setcounter { enumXvi } { 0 }
3127        \vspace { \l__enumext_topsep_v_skip }
3128        %\bool_set_false:N \l__enumext_store_active_bool
3129    }
```

(*End of definition for* keyanspic. *This function is documented on page* 12.)

`\__enumext_keyans_pic_safe_exec:` The function `\__enumext_keyans_pic_safe_exec:` check nested and level position inside the `enumext` environment.

```
3130  \cs_new_protected:Nn \__enumext_keyans_pic_safe_exec:
3131    {
3132        \int_incr:N \l__enumext_keyans_pic_level_int
3133        \int_compare:nNnT { \l__enumext_keyans_pic_level_int } > { 1 }
3134          {
3135            \msg_error:nn { enumext } { keyanspic-nested }
3136          }
3137        \__enumext_keyans_save_start_line:
3138    }
```

(*End of definition for* __enumext_keyans_pic_safe_exec:.)

`\__enumext_keyans_pic_skip_abs:N` The function `\__enumext_keyans_pic_skip_abs:N` will return a positive value `\parsep`.

```
3139  \cs_new_protected:Npn \__enumext_keyans_pic_skip_abs:N #1
3140    {
3141        \dim_compare:nNnT { #1 } < { 0pt }
3142          { \skip_set:Nn #1 { -#1 } }
3143    }
```

(*End of definition for* __enumext_keyans_pic_skip_abs:N.)

`\__enumext_keyans_pic_arg_two:` The function `\__enumext_keyans_pic_arg_two:` will be used in the second argument of the `\__enumext_start_list:nn` function that defines the `keyanspic` environment, it will handle the setting of spaces.

```
3144  \cs_new_protected:Nn \__enumext_keyans_pic_arg_two:
3145    {
```

The first thing to do is to set the boolean variable `\l__enumext_leftmargin_tmp_v_bool` handled by the `list-indent` key to false, then we copy the definition of the second list argument from the `keyans` environment.

```
3146        \bool_set_false:N \l__enumext_leftmargin_tmp_v_bool
3147        \__enumext_list_arg_two_v:
```

We will add the value of `\itemsep` to `\parsep` which we will use as vertical spacing between the above and below `minipage` environments. and adjust the value of `\leftmargin`, the label and counter are handled directly by the `\anspic` command. Then we make equal to zero `\labelwidth`, `\labelsep`, `\partopsep` and `\itemsep` so that the horizontal and vertical spacing is not affected.

```
3148        \skip_add:Nn \parsep { \itemsep }
3149        \dim_add:Nn  \leftmargin { -\labelwidth - \labelsep }
3150        \dim_zero:N  \labelwidth
3151        \dim_zero:N  \listparindent
3152        \dim_zero:N  \labelsep
3153        \skip_zero:N \partopsep
3154        \skip_zero:N \itemsep
```

We set the value of \l__enumext_keyans_pic_above_skip which we will use to apply our "adjust" space above keyanspic, finally we call __enumext_item_std:w followed by \scan_stop: to prevent the error message returned by LATEX when not using the \item command.

```
3155     \__enumext_keyans_pic_skip_abs:N \parsep
3156     \skip_set:Nn \l__enumext_keyans_pic_above_skip
3157       {
3158         \box_dp:N \strutbox
3159         + \l__enumext_topsep_v_skip
3160         - \parsep
3161       }
3162     \__enumext_item_std:w \scan_stop:
3163   }
```

(*End of definition for* __enumext_keyans_pic_arg_two:.)

__enumext_keyans_pic_do:n
__enumext_keyans_pic_do:e

The optional argument is split by comma and is handled directly by the function __enumext_keyans_-pic_do:n and passed to the function __enumext_keyans_pic_row:n.

```
3164 \cs_new_protected:Nn \__enumext_keyans_pic_do:n
3165   {
3166     \clist_map_function:nN { #1 } \__enumext_keyans_pic_row:n
3167   }
3168 \cs_generate_variant:Nn \__enumext_keyans_pic_do:n { e }
```

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

__enumext_keyans_pic_row:n

The function __enumext_keyans_pic_row:n will set the widths for the minipage environments and place the content ⟨*stored*⟩ by \anspic* in the \l__enumext_keyans_pic_body_seq sequence inside them.

```
3169 \cs_new_protected:Nn \__enumext_keyans_pic_row:n
3170   {
3171     \dim_set:Nn \l__enumext_keyans_pic_width_dim { \linewidth / #1 }
3172     \int_set:Nn \l__enumext_keyans_pic_above_int { \l__enumext_keyans_pic_below_int }
3173     \int_set:Nn \l__enumext_keyans_pic_below_int { \l__enumext_keyans_pic_above_int + #1 }
3174     \int_step_inline:nnn
3175       { \l__enumext_keyans_pic_above_int + 1 }
3176       { \l__enumext_keyans_pic_below_int }
3177       {
3178         \__enumext_minipage:w [ b ]{ \l__enumext_keyans_pic_width_dim }
3179           \centering
3180           \seq_item:Nn \l__enumext_keyans_pic_body_seq { ##1 }
3181         \__enumext_endminipage:
3182       }
3183     \par
3184   }
```

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

10.35 The environment enumext*

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 as in the non starred versions (at least I have not achieved it) and as we will make it behave differently, we have no other option than to define a cascade of functions.

To achieve the horizontal list environment we will capture the \item command and the content of this in an plain lrbox 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 optional first argument (⟨*number*⟩).

10.35.1 Functions for item box width

__enumext_starred_columns_set_vii:

We set the default value for the width of the box containing the content of the items and create \itemwidth in a public form.

```
3185 \cs_new_protected:Nn \__enumext_starred_columns_set_vii:
3186   {
3187     \dim_compare:nNnT { \l__enumext_columns_sep_vii_dim } = { \c_zero_dim }
3188       {
3189         \dim_set:Nn \l__enumext_columns_sep_vii_dim
3190           {
3191             ( \l__enumext_labelwidth_vii_dim + \l__enumext_labelsep_vii_dim )
```

```
3192                 / \l__enumext_columns_vii_int
3193               }
3194           }
3195       \int_set:Nn \l__enumext_tmpa_vii_int { \l__enumext_columns_vii_int - \c_one_int }
3196       \dim_set:Nn \l__enumext_item_width_vii_dim
3197         {
3198           ( \linewidth - \l__enumext_columns_sep_vii_dim * \l__enumext_tmpa_vii_int )
3199           / \l__enumext_columns_vii_int - \l__enumext_labelwidth_vii_dim
3200           - \l__enumext_labelsep_vii_dim
3201         }
3202       \dim_zero_new:N \itemwidth
3203     }
```

(*End of definition for* __enumext_starred_columns_set_vii:.)

__enumext_starred_joined_item_vii:n   The function __enumext_starred_joined_item_vii:n will set the *width* of the box in which the content passed to \item(⟨*number*⟩) will be stored together with the value of \itemwidth.

```
3204  \cs_new_protected:Npn \__enumext_starred_joined_item_vii:n #1
3205    {
3206      \int_set:Nn \l__enumext_joined_item_vii_int {#1}
3207      \int_compare:nNnT { \l__enumext_joined_item_vii_int } > { \l__enumext_columns_vii_int }
3208        {
3209          \msg_warning:nnee { enumext } { item-joined }
3210            { \int_use:N \l__enumext_joined_item_vii_int }
3211            { \int_use:N \l__enumext_columns_vii_int }
3212          \int_set:Nn \l__enumext_joined_item_vii_int
3213            {
3214              \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + \c_one_int
3215            }
3216        }
3217      \int_compare:nNnT
3218        { \l__enumext_joined_item_vii_int }
3219          >
3220        { \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + \c_one_int }
3221        {
3222          \msg_warning:nnee { enumext } { item-joined-columns }
3223            { \int_use:N \l__enumext_joined_item_vii_int }
3224            {
3225              \int_eval:n
3226                { \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + \c_one_int }
3227            }
3228          \int_set:Nn \l__enumext_joined_item_vii_int
3229            {
3230              \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + \c_one_int
3231            }
3232        }
```

Only need if #1 » 1 (default are set before).

```
3233      \int_compare:nNnTF { \l__enumext_joined_item_vii_int } > { \c_one_int }
3234        {
3235          \int_set_eq:NN \l__enumext_joined_item_aux_vii_int \l__enumext_joined_item_vii_int
3236          \int_decr:N \l__enumext_joined_item_aux_vii_int
3237          \int_add:Nn \l__enumext_item_column_pos_vii_int { \l__enumext_joined_item_aux_vii_int }
3238          \int_gadd:Nn \g__enumext_item_count_all_vii_int { \l__enumext_joined_item_aux_vii_int }
3239          \dim_set:Nn \l__enumext_joined_width_vii_dim
3240            {
3241              \l__enumext_item_width_vii_dim * \l__enumext_joined_item_vii_int
3242              + (  \l__enumext_labelwidth_vii_dim + \l__enumext_labelsep_vii_dim
3243                 + \l__enumext_columns_sep_vii_dim
3244                )*\l__enumext_joined_item_aux_vii_int
3245            }
3246          \dim_set_eq:NN \itemwidth \l__enumext_joined_width_vii_dim
3247        }
3248        {
3249          \dim_set_eq:NN \l__enumext_joined_width_vii_dim \l__enumext_item_width_vii_dim
3250          \dim_set_eq:NN \itemwidth \l__enumext_item_width_vii_dim
3251        }
3252    }
```

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

__enumext_start_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_env*` 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`.

```
3253  \cs_new_protected:Nn \__enumext_start_mini_vii:
3254    {
3255      \dim_compare:nNnT { \l__enumext_minipage_right_vii_dim } > { \c_zero_dim }
3256        {
3257          \dim_set:Nn \l__enumext_minipage_left_vii_dim
3258            {
3259              \linewidth
3260              - \l__enumext_minipage_right_vii_dim
3261              - \l__enumext_minipage_hsep_vii_dim
3262            }
3263          \bool_set_true:N \l__enumext_minipage_active_vii_bool
3264          \dim_gset_eq:NN
3265            \g__enumext_minipage_right_vii_dim
3266            \l__enumext_minipage_right_vii_dim
3267          \__enumext_mini_addvspace_vii:
3268          \nointerlineskip\noindent
3269          \begin{__enumext_mini_env*}{ \l__enumext_minipage_left_vii_dim }
3270        }
3271    }
```

(*End of definition for* __enumext_start_mini_vii:.)

__enumext_stop_mini_vii:

The function `\__enumext_stop_mini_vii:` closes the `__enumext_mini_env*` environment on the left side, applies `\hfill` and sets the value of the variable `\g__enumext_minipage_active_vii_bool` to true which will be used in the function `\__enumext_after_star_env:nn` to execute the `__enumext_-mini_env*` on the *"right side"*.

```
3272  \cs_new_protected:Nn \__enumext_stop_mini_vii:
3273    {
3274      \bool_if:NT \l__enumext_minipage_active_vii_bool
3275        {
3276          \end{__enumext_mini_env*}
3277          \hfill
3278          \bool_gset_true:N \g__enumext_minipage_active_vii_bool
3279        }
3280    }
```

Finally we execute code passed to the `miniright` key stored in the variable `\g__enumext_miniright_-code_vii_tl` in the `__enumext_mini_env*` environment on the *"right side"*.

```
3281  \__enumext_after_env:nn {enumext*}
3282    {
3283      \bool_if:NT \g__enumext_minipage_active_vii_bool
3284        {
3285          \begin{__enumext_mini_env*}{ \g__enumext_minipage_right_vii_dim }
3286            \par\addvspace { \g__enumext_minipage_right_skip }
3287            \bool_if:NF \g__enumext_minipage_center_vii_bool
3288              {
3289                \centering
3290              }
3291            \tl_use:N \g__enumext_miniright_code_vii_tl % the code
3292          \end{__enumext_mini_env*}
3293          \par\addvspace{ \g__enumext_minipage_after_skip }
3294        }
3295      \bool_gset_false:N \g__enumext_minipage_active_vii_bool
3296      \bool_gset_true:N \g__enumext_minipage_center_vii_bool
3297      \tl_gclear:N \g__enumext_miniright_code_vii_tl
3298      \dim_gzero:N \g__enumext_minipage_right_vii_dim
3299      \bool_gset_false:N \g__enumext_starred_bool
3300    }
```

(*End of definition for* __enumext_stop_mini_vii:.)

enumext*

First we will generate the environment and we will give a temporary definition to `\__enumext_stop_-item_tmp_vii:` equal to `\noindent` and next to `\item` equal to `\__enumext_start_item_tmp_vii:` which we will redefine later.

```
3301  \NewDocumentEnvironment{enumext*}{ o }
3302    {
```

```
3303     \__enumext_safe_exec_vii:
3304     \__enumext_parse_keys_vii:n {#1}
3305     \__enumext_before_list_vii:
3306     \__enumext_start_store_level_vii:
3307     \__enumext_start_list:nn { }
3308       {
3309         \__enumext_list_arg_two_vii:
3310         \__enumext_before_keys_exec_vii:
3311       }
3312     \__enumext_starred_columns_set_vii:
3313     \item[] \scan_stop:
3314     \cs_set_eq:NN \__enumext_stop_item_tmp_vii: \noindent
3315     \cs_set_eq:NN \item \__enumext_start_item_tmp_vii:
3316   }
3317   {
3318     \__enumext_stop_item_tmp_vii:
3319     \__enumext_remove_extra_parsep_vii:
3320     \__enumext_stop_list:
3321     \__enumext_stop_store_level_vii:
3322     \__enumext_after_list_vii:
3323   }
```

(*End of definition for* enumext*. *This function is documented on page* 4.)

__enumext_safe_exec_vii: First check the maximum nesting level for the enumext* environment then set the vars \l__enumext_-starred_bool and \g__enumext_starred_bool.

```
3324   \cs_new_protected:Nn \__enumext_safe_exec_vii:
3325     {
3326       \__enumext_is_not_nested:
3327       \int_incr:N \l__enumext_level_h_int
3328       \int_compare:nNnT { \l__enumext_level_h_int } > { 1 }
3329         {
3330           \msg_error:nn { enumext } { nested }
3331         }
3332       \bool_set_true:N \l__enumext_starred_bool
3333       \__enumext_is_on_first_level:
3334     }
```

(*End of definition for* __enumext_safe_exec_vii:.)

__enumext_parse_keys_vii:n Parse [⟨*key* = *val*⟩] for enumext*. If the variable \l__enumext_store_active_bool is true it will call the function __enumext_parse_store_keys_vii:n and reprocess the keys to pass them to the storage sequence.

```
3335   \cs_new_protected:Npn \__enumext_parse_keys_vii:n #1
3336     {
3337       \tl_if_novalue:nF {#1}
3338         {
3339           \str_clear:N \l__enumext_series_str
3340           \keys_set:nn { enumext / enumext* } {#1}
3341           \__enumext_parse_series:n {#1}
3342           \bool_if:NT \l__enumext_store_active_bool
3343             {
3344               \__enumext_parse_store_keys_vii:n {#1}
3345             }
3346         }
3347     }
```

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

__enumext_parse_store_keys_vii:n The function __enumext_parse_store_keys_vii:n searches for the values of the columns and columns-sep keys in the optional argument in enumext* environment as long as the starred versions of the columns* and columns-sep* keys are not active. The captured values are stored in the variable \l__enumext_store_opt_vii_tl which is used by the function __enumext_store_level_open_-vii:.

```
3348   \cs_new_protected:Npn \__enumext_parse_store_keys_vii:n #1
3349     {
3350       \bool_if:NF \l__enumext_store_columns_vii_bool
3351         {
3352           \regex_match:nnT { \b columns\b } {#1}
3353             {
```

```
3354        \int_set_eq:NN
3355          \l__enumext_store_columns_vii_int
3356          \l__enumext_columns_vii_int
3357        \tl_put_right:Ne \l__enumext_store_opt_vii_tl
3358          {
3359            columns = \exp_not:V \l__enumext_store_columns_vii_int ,
3360          }
3361        }
3362      }
3363    \bool_if:NF \l__enumext_store_columns_sep_vii_bool
3364      {
3365        \regex_match:nnT { \b columns-sep \b} {#1}
3366          {
3367            \dim_set_eq:NN
3368              \l__enumext_store_columns_sep_vii_dim
3369              \l__enumext_columns_sep_vii_dim
3370            \tl_put_right:Ne \l__enumext_store_opt_vii_tl
3371              {
3372                columns-sep = \exp_not:V \l__enumext_store_columns_sep_vii_dim,
3373              }
3374          }
3375      }
3376  }
```

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

__enumext_before_list_vii: The function __enumext_before_list_vii: 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_vii: handle by mini-env.

```
3377 \cs_new_protected:Nn \__enumext_before_list_vii:
3378   {
3379     \__enumext_vspace_above_vii:
3380     \__enumext_check_ans: % need by chek-ans
3381     \__enumext_before_args_exec_vii:
3382     \__enumext_start_mini_vii:
3383   }
```

(*End of definition for* __enumext_before_list_vii:.)

__enumext_after_list_vii: The function __enumext_after_list: first call the function __enumext_stop_mini_vii:, then apply the {⟨code⟩} handled by the after key together with the *vertical space* handled by the below key if they are present. Finally set false the vars \g__enumext_starred_bool and \l__enumext_starred_- bool, save the *current value* of the counter in \g__enumext_resume_vii_int for the resume key. If the save-ans key is active, it will create the integer variable for the resume key, we only have to assign it the value of the current counter.

```
3384 \cs_new_protected:Nn \__enumext_after_list_vii:
3385   {
3386     \__enumext_stop_mini_vii:
3387     \__enumext_after_stop_list_vii:
3388     \__enumext_check_ans_to_hook:
3389     \__enumext_vspace_below_vii:
3390     \bool_set_false:N \l__enumext_starred_bool
3391     \__enumext_resume_save_counter:
3392   }
```

(*End of definition for* __enumext_after_list_vii:.)

__enumext_start_store_level_vii:
__enumext_stop_store_level_vii:
The __enumext_start_store_level_vii: and __enumext_stop_store_level_vii: functions activate the level saving mechanism for storage in ⟨sequence⟩ of the \anskey command if enumext* are nested in enumext.

```
3393 \cs_new_protected:Nn \__enumext_start_store_level_vii:
3394   {
3395     \bool_if:NT \l__enumext_store_active_bool
3396       {
3397         \int_compare:nNnT { \l__enumext_level_int } > { \c_zero_int }
3398           {
3399             \__enumext_store_level_open_vii:
3400           }
3401       }
3402   }
```

```
3403 \cs_new_protected:Nn \__enumext_stop_store_level_vii:
3404   {
3405     \bool_if:NT \l__enumext_store_active_bool
3406       {
3407         \int_compare:nNnT { \l__enumext_level_int } > { \c_zero_int }
3408           {
3409             \__enumext_store_level_close_vii:
3410           }
3411       }
3412   }
```

(*End of definition for* __enumext_start_store_level_vii: *and* __enumext_stop_store_level_vii:*.*)

10.35.2 The command \item in enumext*

__enumext_start_item_tmp_vii: First we will call the function __enumext_stop_item_tmp_vii: that we will redefine later, we will increment the value of \l__enumext_item_column_pos_vii_int that will count the item's by rows and the value of \g__enumext_item_count_all_vii_int that will count the total of item's in the environment. After that we will call the function __enumext_item_peek_args_vii: that will handle the arguments passed to \item.

```
3413 \cs_new_protected_nopar:Nn \__enumext_start_item_tmp_vii:
3414   {
3415     \__enumext_stop_item_tmp_vii:
3416     \int_incr:N \l__enumext_item_column_pos_vii_int
3417     \int_gincr:N \g__enumext_item_count_all_vii_int
3418     \__enumext_item_peek_args_vii:
3419   }
```

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

__enumext_item_peek_args_vii: The function __enumext_item_peek_args_vii: will handle the \item(⟨*number*⟩). Look for the argument "(", if it is present we will call the function __enumext_joined_item_vii:w (⟨*number*⟩), which is in charge of joining the item's in the same row, in case they are not present we will set the default value (1).

```
3420 \cs_new_protected:Nn \__enumext_item_peek_args_vii:
3421   {
3422     \peek_meaning:NTF (
3423       { \__enumext_joined_item_vii:w }
3424       { \__enumext_joined_item_vii:w (1) }
3425   }
```

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

__enumext_joined_item_vii:w   The function __enumext_joined_item_vii:w will first call the function __enumext_starred_-joined_item_vii:n in charge of setting the *width* of the box that will store the content passed to \item. Then we will look for the argument "*", if it is present we will call the function __enumext_starred_-item_vii:w otherwise we will call the function __enumext_standard_item_vii:w.

```
3426 \cs_new_protected:Npn \__enumext_joined_item_vii:w (#1)
3427   {
3428     \__enumext_starred_joined_item_vii:n {#1}
3429     \peek_meaning_remove:NTF *
3430       { \__enumext_starred_item_vii:w  }
3431       { \__enumext_standard_item_vii:w }
3432   }
```

(*End of definition for* __enumext_joined_item_vii:w*.*)

__enumext_standard_item_vii:w   The function __enumext_standard_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].

```
3433 \cs_new_protected:Npn \__enumext_standard_item_vii:w
3434   {
3435     \bool_set_false:N \l__enumext_item_starred_vii_bool
3436       \peek_meaning:NTF [
3437         {
3438           \bool_set_eq:NN
```

```
3439                    \l__enumext_wrap_label_vii_bool
3440                    \l__enumext_wrap_label_opt_vii_bool
3441                  \__enumext_start_item_vii:w
3442              }
3443              {
3444                  \bool_set_true:N \l__enumext_wrap_label_vii_bool
3445                  \legacy_if_set_true:n { @noitemarg }
3446                  \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
3447              }
3448          }
```

(*End of definition for* `\__enumext_standard_item_vii:w`.)

`\__enumext_starred_item_vii:w`
`\__enumext_starred_item_vii_aux_i:w`
`\__enumext_starred_item_vii_aux_ii:w`
`\__enumext_starred_item_vii_aux_iii:w`

The function `\__enumext_starred_item_vii:w` together with the specified auxiliary functions `aux_i:w`, `aux_ii:w`, and `aux_iii:w` execute `\item*`, `\item*[`⟨*symbol*⟩`]` and `\item*[`⟨*symbol*⟩`][`⟨*offset*⟩`]`.

```
3449  \cs_new_protected:Npn \__enumext_starred_item_vii:w
3450    {
3451      \bool_set_true:N \l__enumext_item_starred_vii_bool
3452      \bool_set_true:N \l__enumext_wrap_label_vii_bool
3453      \peek_meaning:NTF [
3454        { \__enumext_starred_item_vii_aux_i:w }
3455        { \__enumext_starred_item_vii_aux_ii:w }
3456    }
3457  \cs_new_protected:Npn \__enumext_starred_item_vii_aux_i:w [#1]
3458    {
3459      \tl_gset:Nn \g__enumext_item_symbol_aux_vii_tl {#1}
3460      \__enumext_starred_item_vii_aux_ii:w
3461    }
3462  \cs_new_protected:Npn \__enumext_starred_item_vii_aux_ii:w
3463    {
3464      \peek_meaning:NTF [
3465        { \__enumext_starred_item_vii_aux_iii:w }
3466        {
3467          \dim_set_eq:NN
3468            \l__enumext_item_symbol_sep_vii_dim
3469            \l__enumext_labelsep_vii_dim
3470          \legacy_if_set_true:n { @noitemarg }
3471          \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
3472        }
3473    }
3474  \cs_new_protected:Npn \__enumext_starred_item_vii_aux_iii:w [#1]
3475    {
3476      \dim_set:Nn \l__enumext_item_symbol_sep_vii_dim {#1}
3477      \legacy_if_set_true:n { @noitemarg }
3478      \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
3479    }
```

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

10.35.3   Real definition of `\item` in enumext*

`\__enumext_start_item_vii:w`

The functions `\__enumext_start_item_vii:w` and `\__enumext_stop_item_vii:` executing the true definition of `\item` inside the enumext* environment.

The first thing we will do is set the value of `\__enumext_stop_item_tmp_vii:` equal to the value of `\__enumext_stop_item_vii:` which we will define later and add the hyperref compatible enumXvii counter, after that we will start capturing the item content in a box. Here need setting the `\if@hyper@item` switch to *"true"* for hyperref compatible. The explanation for this is given by the master Heiko Oberdiek on \refstepcounter{enumi} twice (or more) creates destination with the same identifier.

```
3480  \cs_new_protected_nopar:Npn \__enumext_start_item_vii:w [#1]
3481    {
3482      \cs_set_eq:NN \__enumext_stop_item_tmp_vii: \__enumext_stop_item_vii:
3483      \legacy_if:nT { @noitemarg }
3484        {
3485          \legacy_if_set_false:n { @noitemarg }
3486          \legacy_if:nT { @nmbrlist }
3487            {
3488              \bool_if:NT \l__enumext_hyperref_bool
3489                {
3490                  \legacy_if_set_true:n { @hyper@item }
3491                }
3492              \refstepcounter{enumXvii}
```

```
3493            \bool_if:NT \l__enumext_check_ans_bool
3494              {
3495                \int_gincr:N \g__enumext_count_item_number_int
3496              }
3497          }
3498        }
```

Here we start capturing \item and its contents into a group using the plain form of the lrbox environment. If the state of the variable \l__enumext_footnotes_key_bool is false, we will redefine the command \footnote, followed by printing the ⟨symbol⟩ defined for \item* if it is present and open a new group inside which we execute font key next to \item and the keys wrap-label, wrap-label*, align, close the group and execute the key labelsep and then the key first. Finally we open the minipage environment and execute the listparindent key which will be equal to \parindent, the parsep key which will be equal to \parskip and the itemindent key.

```
3499        \group_begin:
3500          \lrbox{ \l__enumext_item_text_vii_box }
3501            \bool_if:NF \l__enumext_footnotes_key_bool
3502              {
3503                \__enumext_renew_footnote:
3504              }
3505            \bool_if:NT \l__enumext_item_starred_vii_bool
3506              {
3507                \tl_if_blank:VT \g__enumext_item_symbol_aux_vii_tl
3508                  {
3509                    \tl_gset_eq:NN
3510                      \g__enumext_item_symbol_aux_vii_tl \l__enumext_item_symbol_vii_tl
3511                  }
3512                \mode_leave_vertical:
3513                \skip_horizontal:n { -\l__enumext_item_symbol_sep_vii_dim }
3514                \makebox[ 0pt ][ r ]{ \g__enumext_item_symbol_aux_vii_tl }
3515                \skip_horizontal:N \l__enumext_item_symbol_sep_vii_dim
3516                \tl_gclear:N \g__enumext_item_symbol_aux_vii_tl
3517              }
3518          \group_begin:
3519            \tl_use:N \l__enumext_label_font_style_vii_tl
3520            \bool_if:NTF \l__enumext_wrap_label_vii_bool
3521              {
3522                \makebox[ \l__enumext_labelwidth_vii_dim ][ \l__enumext_align_label_vii_str ]
3523                  { \__enumext_wrapper_label_vii:n {#1} }
3524              }
3525              {
3526                \makebox[ \l__enumext_labelwidth_vii_dim ][ \l__enumext_align_label_vii_str ]{ #1 }
3527              }
3528          \group_end:
3529          \skip_horizontal:N \l__enumext_labelsep_vii_dim
3530          \tl_use:N \l__enumext_after_list_args_vii_tl
3531          \__enumext_minipage:w [ t ]{ \l__enumext_joined_width_vii_dim  }
3532            \skip_set_eq:NN \parindent \l__enumext_listparindent_vii_dim
3533            \skip_set_eq:NN \parskip \l__enumext_parsep_vii_skip
3534            \tl_use:N \l__enumext_fake_item_indent_vii_tl
3535        }
```

(*End of definition for* __enumext_start_item_vii:w.)

__enumext_stop_item_vii: The function __enumext_stop_item_vii: shall terminate with the capture of \item and its ⟨contents⟩. Close the environments minipage, lrbox and the group. Then we only have to set the width of the box and print it next to \footnote, and add the horizontal and vertical separation between the boxes.

```
3536  \cs_new_protected_nopar:Nn \__enumext_stop_item_vii:
3537    {
3538        \__enumext_endminipage:
3539      \endlrbox
3540    \group_end:
3541    \box_set_wd:Nn \l__enumext_item_text_vii_box
3542      {
3543        \l__enumext_joined_width_vii_dim
3544        + \l__enumext_labelwidth_vii_dim
3545        + \l__enumext_labelsep_vii_dim
3546      }
3547    \int_set:Nn \hbadness { 10000 }
3548    \box_use:N \l__enumext_item_text_vii_box
3549    \bool_if:NF \l__enumext_footnotes_key_bool
```

```
3550        {
3551            \__enumext_print_footnote:
3552        }
3553    \int_compare:nNnTF { \l__enumext_item_column_pos_vii_int } = { \l__enumext_columns_vii_int }
3554        {
3555            \par\noindent
3556            \int_zero:N \l__enumext_item_column_pos_vii_int
3557        }
3558        { \hspace{ \l__enumext_columns_sep_vii_dim } }
3559    }
```

(*End of definition for* __enumext_stop_item_vii:.)

__enumext_remove_extra_parsep_vii: Finally we will remove the vertical space equal to \parsep when the total number of items is divisible by the number of items in the last row of the environment.

```
3560 \cs_new_protected:Nn \__enumext_remove_extra_parsep_vii:
3561    {
3562    \int_compare:nNnT
3563        {
3564            \int_mod:nn {  \g__enumext_item_count_all_vii_int } { \l__enumext_columns_vii_int }
3565        }
3566        =
3567        { \c_zero_int }
3568        {
3569            \par
3570            \vspace{ -\l__enumext_itemsep_vii_skip }
3571            \int_gzero:N \g__enumext_item_count_all_vii_int
3572        }
3573    }
```

(*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.

```
3574 \__enumext_after_env:nn {enumext*} { \__enumext_check_ans_show: }
```

10.36   The environment keyans*

10.36.1   Functions for item box width

__enumext_starred_columns_set_viii: We set the default value for the width of the box containing the content of the items and create \itemwidth in a public form.

```
3575 \cs_new_protected:Nn \__enumext_starred_columns_set_viii:
3576    {
3577    \dim_compare:nNnT { \l__enumext_columns_sep_viii_dim } = { \c_zero_dim }
3578        {
3579            \dim_set:Nn \l__enumext_columns_sep_viii_dim
3580            {
3581                ( \l__enumext_labelwidth_viii_dim + \l__enumext_labelsep_viii_dim )
3582                / \l__enumext_columns_viii_int
3583            }
3584        }
3585    \int_set:Nn \l__enumext_tmpa_viii_int { \l__enumext_columns_viii_int - \c_one_int }
3586    \dim_set:Nn \l__enumext_item_width_viii_dim
3587        {
3588            ( \linewidth - \l__enumext_columns_sep_viii_dim * \l__enumext_tmpa_viii_int )
3589            / \l__enumext_columns_viii_int - \l__enumext_labelwidth_viii_dim
3590            - \l__enumext_labelsep_viii_dim
3591        }
3592    \dim_zero_new:N \itemwidth
3593    }
```

(*End of definition for* __enumext_starred_columns_set_viii:.)

__enumext_starred_joined_item_viii:n The function __enumext_starred_joined_item_viii:n will set the *width* of the box in which the content passed to \item(⟨*number*⟩) will be stored together with the value of \itemwidth.

```
3594 \cs_new_protected:Npn \__enumext_starred_joined_item_viii:n #1
3595    {
3596    \int_set:Nn \l__enumext_joined_item_viii_int {#1}
3597    \int_compare:nNnT { \l__enumext_joined_item_viii_int } > { \l__enumext_columns_viii_int }
3598        {
3599            \msg_warning:nnee { enumext } { item-joined }
```

```
3600              { \int_use:N \l__enumext_joined_item_viii_int }
3601              { \int_use:N \l__enumext_columns_viii_int }
3602          \int_set:Nn \l__enumext_joined_item_viii_int
3603            {
3604                \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + \c_one_int
3605            }
3606          }
3607      \int_compare:nNnT
3608        { \l__enumext_joined_item_viii_int }
3609          >
3610        { \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + \c_one_int }
3611        {
3612          \msg_warning:nnee { enumext } { item-joined-columns }
3613            { \int_use:N \l__enumext_joined_item_viii_int }
3614            {
3615              \int_eval:n
3616                { \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + \c_one_int }
3617            }
3618          \int_set:Nn \l__enumext_joined_item_viii_int
3619            {
3620                \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + \c_one_int
3621            }
3622        }
3623      \int_compare:nNnTF { \l__enumext_joined_item_viii_int } > { \c_one_int }
3624        {
3625          \int_set_eq:NN \l__enumext_joined_item_aux_viii_int \l__enumext_joined_item_viii_int
3626          \int_decr:N \l__enumext_joined_item_aux_viii_int
3627          \int_add:Nn \l__enumext_item_column_pos_viii_int { \l__enumext_joined_item_aux_viii_int }
3628          \int_gadd:Nn \g__enumext_item_count_all_viii_int { \l__enumext_joined_item_aux_viii_int }
3629          \dim_set:Nn \l__enumext_joined_width_viii_dim
3630            {
3631              \l__enumext_item_width_viii_dim * \l__enumext_joined_item_viii_int
3632              + (  \l__enumext_labelwidth_viii_dim + \l__enumext_labelsep_viii_dim
3633                + \l__enumext_columns_sep_viii_dim
3634              )*\l__enumext_joined_item_aux_viii_int
3635            }
3636          \dim_set_eq:NN \itemwidth \l__enumext_joined_width_viii_dim
3637        }
3638        {
3639          \dim_set_eq:NN \l__enumext_joined_width_viii_dim \l__enumext_item_width_viii_dim
3640          \dim_set_eq:NN \itemwidth \l__enumext_item_width_viii_dim
3641        }
3642    }
```

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

__enumext_start_mini_viii: The implementation of the `mini-env` key is identical to the one used in the `enumext*` environment.

__enumext_stop_mini_viii:

```
3643  \cs_new_protected:Nn \__enumext_start_mini_viii:
3644    {
3645      \dim_compare:nNnT { \l__enumext_minipage_right_viii_dim } > { \c_zero_dim }
3646        {
3647          \dim_set:Nn \l__enumext_minipage_left_viii_dim
3648            {
3649              \linewidth
3650              - \l__enumext_minipage_right_viii_dim
3651              - \l__enumext_minipage_hsep_viii_dim
3652            }
3653          \bool_set_true:N \l__enumext_minipage_active_viii_bool
3654          \dim_gset_eq:NN
3655            \g__enumext_minipage_right_viii_dim
3656            \l__enumext_minipage_right_viii_dim
3657          \__enumext_mini_addvspace_viii:
3658          \nointerlineskip\noindent
3659          \begin{__enumext_mini_env*}{ \l__enumext_minipage_left_viii_dim }
3660        }
3661    }
3662  \cs_new_protected:Nn \__enumext_stop_mini_viii:
3663    {
3664      \bool_if:NT \l__enumext_minipage_active_viii_bool
3665        {
3666          \end{__enumext_mini_env*}
```

```
3667          \hfill
3668          \bool_gset_true:N \g__enumext_minipage_active_viii_bool
3669        }
3670      }
3671  \__enumext_after_env:nn {keyans*}
3672    {
3673      \bool_if:NT \g__enumext_minipage_active_viii_bool
3674        {
3675          \begin{__enumext_mini_env*}{ \g__enumext_minipage_right_viii_dim }
3676            \par\addvspace { \g__enumext_minipage_right_skip }
3677            \bool_if:NF \g__enumext_minipage_center_viii_bool
3678              {
3679                \centering
3680              }
3681            \tl_use:N \g__enumext_miniright_code_viii_tl % the code
3682          \end{__enumext_mini_env*}
3683          \par\addvspace{ \g__enumext_minipage_after_skip }
3684        }
3685      \bool_gset_false:N \g__enumext_minipage_active_viii_bool
3686      \bool_gset_true:N \g__enumext_minipage_center_viii_bool
3687      \tl_gclear:N \g__enumext_miniright_code_viii_tl
3688      \dim_gzero:N \g__enumext_minipage_right_viii_dim
3689    }
```

(*End of definition for* __enumext_start_mini_viii: *and* __enumext_stop_mini_viii:.)

keyans*   First we will generate the environment and we will give a temporary definition to __enumext_stop_-item_tmp_viii: equal to \noindent and next to \item equal to __enumext_start_item_tmp_-viii: which we will redefine later.

```
3690  \NewDocumentEnvironment{keyans*}{ o }
3691    {
3692      \__enumext_safe_exec_viii:
3693      \__enumext_parse_keys_viii:n {#1}
3694      \__enumext_before_list_viii:
3695      \__enumext_start_list:nn { }
3696        {
3697          \__enumext_list_arg_two_viii:
3698          \__enumext_before_keys_exec_viii:
3699        }
3700      \__enumext_starred_columns_set_viii:
3701      \item[] \scan_stop:
3702      \cs_set_eq:NN \__enumext_stop_item_tmp_viii: \noindent
3703      \cs_set_eq:NN \item \__enumext_start_item_tmp_viii:
3704    }
3705    {
3706      \__enumext_stop_item_tmp_viii:
3707      \__enumext_remove_extra_parsep_viii:
3708      \__enumext_check_starred_cmd:n { item }
3709      \__enumext_stop_list:
3710      \__enumext_after_list_viii:
3711    }
```

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

__enumext_safe_exec_viii: First check the maximum nesting level for the keyans* environment.

```
3712  \cs_new_protected:Nn \__enumext_safe_exec_viii:
3713    {
3714      \int_incr:N \l__enumext_keyans_level_h_int
3715      \int_compare:nNnT { \l__enumext_keyans_level_h_int } > { 1 }
3716        {
3717          \msg_error:nn { enumext } { nested }
3718        }
3719      \__enumext_keyans_save_start_line:
3720      % Set false for interfering with enumext nested in keyans* (yes, its possible and crayze)
3721      \bool_set_false:N \l__enumext_store_active_bool
3722      \int_compare:nNnT { \l__enumext_level_int } > { 1 }
3723        {
3724          \msg_error:nn { enumext } { keyans-wrong-level }
3725        }
3726    }
```

(*End of definition for* __enumext_safe_exec_viii:.)

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

```
3727 \cs_new_protected:Npn \__enumext_parse_keys_viii:n #1
3728   {
3729     \tl_if_novalue:nF {#1}
3730       {
3731         \keys_set:nn { enumext / keyans* } {#1}
3732       }
3733   }
```

(*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.

```
3734 \cs_new_protected:Nn \__enumext_before_list_viii:
3735   {
3736     \__enumext_vspace_above_viii:
3737     \__enumext_before_args_exec_viii:
3738     \__enumext_start_mini_viii:
3739   }
```

(*End of definition for* __enumext_before_list_viii:.)

__enumext_after_list_viii: The function __enumext_after_list: 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.

```
3740 \cs_new_protected:Nn \__enumext_after_list_viii:
3741   {
3742     \__enumext_stop_mini_viii:
3743     \__enumext_after_stop_list_viii:
3744     \__enumext_vspace_below_viii:
3745   }
```

(*End of definition for* __enumext_after_list_viii:.)

10.36.2   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_start_item_tmp_viii: First we will call the function __enumext_stop_item_tmp_viii: that we will redefine later, we will increment the value of \l__enumext_item_column_pos_viii_int that will count the item's by rows and the value of \g__enumext_item_count_all_viii_int that will count the total of item's in the environment. After that we will call the function __enumext_item_peek_args_viii: that will handle the arguments passed to \item.

```
3746 \cs_new_protected_nopar:Nn \__enumext_start_item_tmp_viii:
3747   {
3748     \__enumext_stop_item_tmp_viii:
3749     \int_incr:N \l__enumext_item_column_pos_viii_int
3750     \int_gincr:N \g__enumext_item_count_all_viii_int
3751     \__enumext_item_peek_args_viii:
3752   }
```

(*End of definition for* __enumext_start_item_tmp_viii:.)

__enumext_item_peek_args_viii: The function __enumext_item_peek_args_viii: will handle the \item(⟨*number*⟩). Look for the argument "(", if it is present we will call the function __enumext_joined_item_viii:w (⟨*number*⟩), which is in charge of joining the item's in the same row, in case they are not present we will set the default value (1).

```
3753 \cs_new_protected:Nn \__enumext_item_peek_args_viii:
3754   {
3755     \peek_meaning:NTF (
3756       { \__enumext_joined_item_viii:w }
3757       { \__enumext_joined_item_viii:w (1) }
3758   }
```

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

__enumext_joined_item_viii:w

The function __enumext_joined_item_viii:w will first call the function __enumext_starred_-joined_item_viii:n in charge of setting the *width* of the box that will store the content passed to \item. Then we will look for the argument "*", if it is present we will call the function __enumext_starred_-item_viii:w otherwise we will call the function __enumext_standard_item_viii:w.

```
3759  \cs_new_protected:Npn \__enumext_joined_item_viii:w (#1)
3760    {
3761      \__enumext_starred_joined_item_viii:n {#1}
3762      \peek_meaning_remove:NTF *
3763        { \__enumext_starred_item_viii:w  }
3764        { \__enumext_standard_item_viii:w }
3765    }
```

(*End of definition for* __enumext_joined_item_viii:w.)

__enumext_standard_item_viii:w

The function __enumext_standard_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].

```
3766  \cs_new_protected:Npn \__enumext_standard_item_viii:w
3767    {
3768      \bool_set_false:N \l__enumext_item_starred_viii_bool
3769        \peek_meaning:NTF [
3770          {
3771            \bool_set_eq:NN
3772              \l__enumext_wrap_label_viii_bool
3773              \l__enumext_wrap_label_opt_viii_bool
3774            \__enumext_start_item_viii:w
3775          }
3776          {
3777            \bool_set_true:N \l__enumext_wrap_label_viii_bool
3778            \legacy_if_set_true:n { @noitemarg }
3779            \__enumext_start_item_viii:w [ \l__enumext_label_viii_tl ]
3780          }
3781    }
```

(*End of definition for* __enumext_standard_item_viii:w.)

__enumext_starred_item_viii:w
__enumext_starred_item_viii_aux_i:w
__enumext_starred_item_viii_aux_ii:w

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⟩].

```
3782  \cs_new_protected:Npn \__enumext_starred_item_viii:w
3783    {
3784      \bool_set_true:N \l__enumext_item_starred_viii_bool
3785      \bool_set_true:N \l__enumext_wrap_label_viii_bool
3786      \peek_meaning:NTF [
3787        { \__enumext_starred_item_viii_aux_i:w }
3788        { \__enumext_starred_item_viii_aux_ii:w }
3789    }
```

The function __enumext_starred_item_viii_aux_i:w will save the optional argument to \item* in \l__enumext_keyans_item_opt_tl and will save this argument along with the spacing set by the key save-sep in variable \l__enumext_store_keyans_label_tl if present, then call the function __enumext_starred_item_viii_aux_ii:w.

```
3790  \cs_new_protected:Npn \__enumext_starred_item_viii_aux_i:w [#1]
3791    {
3792      \tl_clear:N \l__enumext_store_keyans_label_tl
3793      \tl_if_novalue:nF { #1 }
3794        {
3795          \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_tl
3796            {
3797              \tl_put_right:Ne \l__enumext_store_keyans_label_tl { \l__enumext_store_keyans_item_op
3798              \tl_put_right:Ne \l__enumext_store_keyans_label_tl { #1 }
3799            }
3800          \tl_set:Ne \l__enumext_keyans_item_opt_tl { #1 }
3801        }
```

```
3802        \__enumext_starred_item_viii_aux_ii:w
3803    }
3804  \cs_new_protected:Npn \__enumext_starred_item_viii_aux_ii:w
3805    {
3806      \legacy_if_set_true:n { @noitemarg }
3807      \__enumext_start_item_viii:w [ \l__enumext_label_viii_tl ]
3808    }
```

(*End of definition for* __enumext_starred_item_viii:w, __enumext_starred_item_viii_aux_i:w, *and* __enumext_-
starred_item_viii_aux_ii:w.)

__enumext_starred_item_exec: The function __enumext_starred_item_exec: will be in charge of storing the current ⟨*label*⟩ for
\item* followed by the [⟨*content*⟩] for \item*[⟨*content*⟩] if present in the ⟨*sequence*⟩ and ⟨*prop list*⟩
set by the save-ans key. In this same function the keys show-ans, show-pos and save-ref are imple-
mented.

```
3809  \cs_new_protected:Nn \__enumext_starred_item_exec:
3810    {
3811      \tl_put_left:Ne \l__enumext_store_keyans_label_tl { \l__enumext_label_viii_tl }
3812      \__enumext_store_addto_prop:V \l__enumext_store_keyans_label_tl
3813      \__enumext_keyans_store_ref:
3814      \tl_put_left:Ne \l__enumext_store_keyans_label_tl { \item }
3815      \__enumext_keyans_addto_seq_link:
3816      \int_gincr:N \g__enumext_check_starred_cmd_int
3817      \bool_if:NT \l__enumext_show_answer_bool
3818        {
3819          \__enumext_print_keyans_box:NN \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
3820        }
3821      \bool_if:NT \l__enumext_show_position_bool
3822        {
3823          \tl_set:Ne \l__enumext_mark_answer_sym_tl
3824            {
3825              \group_begin:
3826                \exp_not:N \normalfont
3827                \exp_not:N \footnotesize [ \int_eval:n
3828                  {
3829                    \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
3830                  }
3831                ]
3832              \group_end:
3833            }
3834          \__enumext_print_keyans_box:NN \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
3835        }
3836    }
```

(*End of definition for* __enumext_starred_item_exec:.)

Real definition of \item **in keyans***

__enumext_start_item_viii:w   The implementation at this point is very similar to that of the enumext* environment.

```
3837  \cs_new_protected_nopar:Npn \__enumext_start_item_viii:w [#1]
3838    {
3839      \cs_set_eq:NN \__enumext_stop_item_tmp_viii: \__enumext_stop_item_viii:
3840      \legacy_if:nT { @noitemarg }
3841        {
3842          \legacy_if_set_false:n { @noitemarg }
3843          \legacy_if:nT { @nmbrlist }
3844            {
3845              \bool_if:NT \l__enumext_hyperref_bool
3846                {
3847                  \legacy_if_set_true:n { @hyper@item }
3848                }
3849              \refstepcounter{enumXviii}
3850            }
3851        }
```

Here we start capturing \item and its contents into a group using the plain form of the lrbox environment.

```
3852      \group_begin:
3853        \lrbox{ \l__enumext_item_text_viii_box }
3854          \bool_if:NF \l__enumext_footnotes_key_bool
3855            {
3856              \__enumext_renew_footnote:
3857            }
```

```
3858              \bool_if:NT \l__enumext_item_starred_viii_bool
3859                {
3860                   \__enumext_starred_item_exec:
3861                }
3862            \group_begin:
3863              \tl_use:N \l__enumext_label_font_style_viii_tl
3864              \bool_if:NTF \l__enumext_wrap_label_viii_bool
3865                {
3866                   \makebox[ \l__enumext_labelwidth_viii_dim ][ \l__enumext_align_label_viii_str ]
3867                     { \__enumext_wrapper_label_viii:n {#1} }
3868                }
3869                {
3870                   \makebox[ \l__enumext_labelwidth_viii_dim ][ \l__enumext_align_label_viii_str ]{ #1
3871                }
3872            \group_end:
3873            \skip_horizontal:N \l__enumext_labelsep_viii_dim
3874            \tl_use:N \l__enumext_after_list_args_viii_tl
3875            \__enumext_minipage:w [ t ]{ \l__enumext_joined_width_viii_dim  }
3876              \skip_set_eq:NN \parindent \l__enumext_listparindent_viii_dim
3877              \skip_set_eq:NN \parskip \l__enumext_parsep_viii_skip
3878              \bool_if:NT \l__enumext_item_starred_viii_bool
3879                {
3880                   \tl_use:N \l__enumext_fake_item_indent_viii_tl
3881                   \__enumext_keyans_show_item_opt: \skip_horizontal:n { -\l__enumext_fake_item_indent_
3882                }
3883                {
3884                   \tl_use:N \l__enumext_fake_item_indent_viii_tl
3885                }
3886        }
```

(*End of definition for* __enumext_start_item_viii:w.)

__enumext_stop_item_viii: The function __enumext_stop_item_viii: shall terminate with the capture of \item and its ⟨*contents*⟩. Close the environments minipage, lrbox and the group. Then we only have to set the width of the box and print it next to \footnote, and add the horizontal and vertical separation between the boxes.

```
3887  \cs_new_protected_nopar:Nn \__enumext_stop_item_viii:
3888    {
3889          \__enumext_endminipage:
3890        \endlrbox
3891      \group_end:
3892      \box_set_wd:Nn \l__enumext_item_text_viii_box
3893        {
3894           \l__enumext_joined_width_viii_dim
3895           + \l__enumext_labelwidth_viii_dim
3896           + \l__enumext_labelsep_viii_dim
3897        }
3898      \int_set:Nn \hbadness { 10000 }
3899      \box_use:N \l__enumext_item_text_viii_box
3900      \bool_if:NF \l__enumext_footnotes_key_bool
3901        {
3902           \__enumext_print_footnote:
3903        }
3904      \int_compare:nNnTF { \l__enumext_item_column_pos_viii_int } = { \l__enumext_columns_viii_int }
3905        {
3906           \par\noindent
3907           \int_zero:N \l__enumext_item_column_pos_viii_int
3908        }
3909        { \hspace{ \l__enumext_columns_sep_viii_dim } }
3910    }
```

(*End of definition for* __enumext_stop_item_viii:.)

__enumext_remove_extra_parsep_viii: Finally we will remove the vertical space equal to \parsep when the total number of items is divisible by the number of items in the last row of the environment.

```
3911  \cs_new_protected:Nn \__enumext_remove_extra_parsep_viii:
3912    {
3913      \int_compare:nNnT
3914        {
3915           \int_mod:nn {  \g__enumext_item_count_all_viii_int } { \l__enumext_columns_viii_int }
3916        }
3917           =
```

```
3918        { \c_zero_int }
3919        {
3920          \par
3921          \vspace{ -\l__enumext_itemsep_viii_skip }
3922          \int_gzero:N \g__enumext_item_count_all_viii_int
3923        }
3924    }
```

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

10.37   The command \getkeyans

\getkeyans   The \getkeyans command takes a mandatory argument of the form {⟨*store name : position*⟩}. Retrieve a *"single"* content stored by \anskey, \anspic* and \item* from ⟨*prop list*⟩ defined by save-ans key.

```
3925  \NewDocumentCommand \getkeyans { m }
3926    {
3927      \exp_args:Ne \__enumext_getkeyans_aux:n
3928        { \tl_to_str:e { \text_expand:n {#1} } }
3929    }
```

(*End of definition for* \getkeyans. *This function is documented on page 13.*)

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

```
3930  \cs_new_protected:Npn \__enumext_getkeyans_aux:n #1
3931    {
3932      \str_if_in:nnTF {#1} { : }
3933        {
3934          \use:e
3935            {
3936              \cs_set:Npn \exp_not:N \__enumext_tmp:w ##1 \c_colon_str ##2 \scan_stop:
3937                { {##1} {##2} }
3938            }
3939          \exp_after:wN \__enumext_getkeyans:nn \__enumext_tmp:w #1 \scan_stop:
3940        }
3941        { \msg_error:nnn { enumext } { missing-colon } {#1} }
3942    }
```

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

__enumext_getkeyans:nn   The internal function __enumext_getkeyans:nn will check for the existence of the ⟨*prop list*⟩, if it does not exist it will return an error message, then it will fetch the content specified by the second ⟨*argument*⟩ from ⟨*prop list*⟩.

```
3943  \cs_new_protected:Npn \__enumext_getkeyans:nn #1 #2
3944    {
3945      \prop_if_exist:cF { g__enumext_#1_prop }
3946        { \msg_error:nnn { enumext } { undefined-storage-anskey } {#1} }
3947      \group_begin:
3948        \prop_item:cn { g__enumext_#1_prop }{#2}
3949      \group_end:
3950    }
```

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

10.38   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 to define a set of ⟨*keys*⟩ with which we will control the options of the different nesting levels for the enumext and enumext* environment by storing the values of these in the token list variables \l__enumext_print_keyans_X_tl.

```
3951  \keys_define:nn  { keyanskey / print }
3952    {
3953      level-1 .code:n    = \tl_put_right:Nn \l__enumext_print_keyans_i_tl
3954                            {
3955                              \setenumext[level,1] {#1} \setenumext[print,1] {#1}
3956                            },
3957      level-1 .initial:n = { label=\arabic*., nosep, columns=2, first=\small, font=\small },
3958      level-2 .code:n    = \tl_put_right:Nn \l__enumext_print_keyans_ii_tl
3959                            {
3960                              \setenumext[level,2] {#1} \setenumext[print,2] {#1}
3961                            },
3962      level-2 .initial:n = { nosep, label=(\alph*), first=\small, font=\small },
```

```
3963      level-3 .code:n      = \tl_put_right:Nn \l__enumext_print_keyans_iii_tl
3964                              {
3965                                \setenumext[level,3] {#1} \setenumext[print,3] {#1}
3966                              },
3967      level-3 .initial:n  = { nosep, label=\roman*., first=\small, font=\small },
3968      level-4 .code:n      = \tl_put_right:Nn \l__enumext_print_keyans_iv_tl
3969                              {
3970                                \setenumext[level,4] {#1} \setenumext[print,4] {#1}
3971                              },
3972      level-4 .initial:n  = { nosep, label=\Alph*., first=\small, font=\small },
3973      level-* .code:n      = \tl_put_right:Nn \l__enumext_print_keyans_vii_tl % starred
3974                              {
3975                                \setenumext[enumext*] {#1} %%\setenumext[print,*] {#1}
3976                              },
3977      level-* .initial:n  = { label=\arabic*., nosep, columns=2, first=\small, font=\small },
3978    }
```

\printkeyans   Create a user command to print *"all stored content"* in ⟨*sequence*⟩ for \anskey, \item* and \anspic*.

```
3979  \NewDocumentCommand \printkeyans { s O{} m }
3980    {
3981      \group_begin:
3982        \tl_use:N \l__enumext_print_keyans_i_tl
3983        \tl_use:N \l__enumext_print_keyans_ii_tl
3984        \tl_use:N \l__enumext_print_keyans_iii_tl
3985        \tl_use:N \l__enumext_print_keyans_iv_tl
3986        \tl_use:N \l__enumext_print_keyans_vii_tl
3987        \__enumext_printkeyans:nnn { #1 } { #2 } { #3 }
3988      \group_end:
3989    }
```

(*End of definition for* \printkeyans. *This function is documented on page 13.*)

__enumext_printkeyans:nnn   The internal function __enumext_printkeyans:nnn will check for the existence of the ⟨*sequence*⟩, if it does not exist it will return an error message, then it will fetch the content specified by the first argument mapping the ⟨*sequence*⟩.
#1 :   starred
#2 :   key-val
#3 :   seq-name

```
3990  \cs_new_protected:Npn \__enumext_printkeyans:nnn #1 #2 #3
3991    {
3992      \seq_if_exist:cTF { g__enumext_#3_seq }
3993        {
3994          \seq_if_empty:cF { g__enumext_#3_seq }
3995            {
3996              %%\seq_show:c { g__enumext_#3_seq }
3997              \bool_if:nTF {#1}
3998                {
3999                  \begin{enumext*}[#2]
4000                    \seq_map_inline:cn { g__enumext_#3_seq } { ##1 }
4001                  \end{enumext*}
4002                }
4003                {
4004                  \begin{enumext}[#2]
4005                    \seq_map_inline:cn { g__enumext_#3_seq } { ##1 }
4006                  \end{enumext}
4007                }
4008            }
4009        }
4010        {
4011          \msg_error:nnn { enumext } { undefined-storage-anskey } {#3}
4012        }
4013    }
```

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

10.39   The command \setenumext

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

```
4014  \keys_define:nn { enumext / meta-families }
4015    {
4016      level-1  .code:n = { \keys_set:nn { enumext / level-1  } {#1} } ,
4017      level-2  .code:n = { \keys_set:nn { enumext / level-2  } {#1} } ,
4018      level-3  .code:n = { \keys_set:nn { enumext / level-3  } {#1} } ,
4019      level-4  .code:n = { \keys_set:nn { enumext / level-4  } {#1} } ,
4020      keyans   .code:n = { \keys_set:nn { enumext / keyans   } {#1} } ,
4021      enumext* .code:n = { \keys_set:nn { enumext / enumext* } {#1} } ,
4022      keyans*  .code:n = { \keys_set:nn { enumext / keyans*  } {#1} } ,
4023      print-1  .code:n = { \keys_set:nn { keyanskey / print } { level-1 = {#1} } } ,
4024      print-2  .code:n = { \keys_set:nn { keyanskey / print } { level-2 = {#1} } } ,
4025      print-3  .code:n = { \keys_set:nn { keyanskey / print } { level-3 = {#1} } } ,
4026      print-4  .code:n = { \keys_set:nn { keyanskey / print } { level-4 = {#1} } } ,
4027      print-*  .code:n = { \keys_set:nn { keyanskey / print } { level-* = {#1} } } ,
4028      unknown  .code:n = { \msg_error:nn { enumext } { unknown-key-family } } ,
4029    }
```

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

```
4030  \seq_const_from_clist:Nn \c__enumext_all_families_seq
4031    {
4032      level-1 , level-2 , level-3 , level-4 , keyans, enumext*,
4033      keyans* , print-1 , print-2 , print-3 , print-4 , print-*,
4034    }
```

\setenumext   Now we define the user command \setenumext.

```
4035  \NewDocumentCommand \setenumext { o +m }
4036    {
4037      \tl_if_novalue:nTF {#1}
4038        {
4039          \seq_map_inline:Nn \c__enumext_all_families_seq
4040        }
4041        {
4042          \seq_clear:N \l__enumext_setkey_tmpa_seq
4043          \seq_set_from_clist:Nn \l__enumext_setkey_tmpb_seq {#1}
4044          \int_set:Nn \l__enumext_setkey_tmpa_int
4045            {
4046              \seq_count:N \l__enumext_setkey_tmpb_seq
4047            }
4048          \int_compare:nNnTF { \l__enumext_setkey_tmpa_int } > { 1 }
4049            {
4050              \seq_pop_left:NN \l__enumext_setkey_tmpb_seq \l__enumext_setkey_tmpa_tl
4051              \seq_map_function:NN \l__enumext_setkey_tmpb_seq \__enumext_set_parse:n
4052              \seq_set_map_e:NNn \l__enumext_setkey_tmpa_seq \l__enumext_setkey_tmpa_seq
4053                {
4054                  \tl_use:N \l__enumext_setkey_tmpa_tl - ##1
4055                }
4056            }
4057            {
4058              \seq_put_right:Ne \l__enumext_setkey_tmpa_seq { \tl_trim_spaces:n {#1} }
4059            }
4060          \seq_if_empty:NTF \l__enumext_setkey_tmpa_seq
4061            { \seq_map_inline:Nn \c__enumext_all_families_seq }
4062            { \seq_map_inline:Nn \l__enumext_setkey_tmpa_seq }
4063        }
4064        {
4065          \keys_set:nn { enumext / meta-families } { ##1 = {#2} }
4066        }
4067    }
```

(End of definition for \setenumext*. This function is documented on page 5.)*

__enumext_set_parse:n   Internal functions used by the \setenumext command.
__enumext_set_error:nn

```
4068  \cs_new_protected:Npn \__enumext_set_parse:n #1
4069    {
4070      \tl_set:Ne \l__enumext_setkey_tmpb_tl { \tl_trim_spaces:n {#1} }
4071      \int_step_inline:nnn { 0 } { 4 } % <- max level
4072        { \tl_remove_all:Nn \l__enumext_setkey_tmpb_tl {##1} }
4073      \tl_if_empty:NTF \l__enumext_setkey_tmpb_tl
```

```
4074        {
4075          \seq_put_right:Ne \l__enumext_setkey_tmpa_seq
4076            { \tl_trim_spaces:n {#1} }
4077        }
4078        { \__enumext_set_error:nn {#1} { } }
4079    }
4080 \cs_new_protected:Npn \__enumext_set_error:nn #1 #2
4081    { \msg_error:nnn { enumext } { invalid-key } {#1} {#2} }
```

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

10.40   Messages

Message used by package-load for multicol and hyperref packages.

```
4082 \msg_new:nnn { enumext } { package-load }
4083    {
4084      The ~ '#1' ~ package ~ is ~ already ~ loaded.
4085    }
4086 \msg_new:nnn { enumext } { package-not-load }
4087    {
4088      The ~ '#1' ~ package ~ will ~ be ~ loaded ~ as ~ a ~ dependency.
4089    }
4090 \msg_new:nnn { enumext } { package-load-foot }
4091    {
4092      The ~ '#1' ~ package ~ is ~ loaded ~ with ~ the ~ option ~ '#2'.
4093    }
```

Message used in the creation of counters by enumext package.

```
4094 \msg_new:nnn { enumext } { counters }
4095    {
4096      The ~ counter ~ '#1' ~ is ~ already ~ defined ~ by ~ some ~ \\
4097      package ~ or ~ macro, ~ it ~ cannot ~ be ~ continued.
4098    }
```

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

```
4099 \msg_new:nnn { enumext } { store-prop }
4100    {
4101      Creating ~ the ~ 'prop~list' ~ '\c_backslash_str g__enumext_#1_prop' ~ \msg_line_context:.
4102    }
4103 \msg_new:nnn { enumext } { store-seq }
4104    {
4105      Creating ~ the ~ 'sequence' ~ '\c_backslash_str g__enumext_#1_seq' ~ \msg_line_context:.
4106    }
4107 \msg_new:nnn { enumext } { store-int }
4108    {
4109      Creating ~ the ~ 'integer' ~ '\c_backslash_str g__enumext_#1_int' ~ \msg_line_context:.
4110    }
```

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

```
4111 \msg_new:nnn { enumext } { invalid-key }
4112    {
4113      The ~ key ~ '#1' ~ is ~ not ~ know ~ the ~ level ~ #2.
4114    }
4115 \msg_new:nnn { enumext } { unknown-key-family }
4116    {
4117      Unknown~key~family~`\l_keys_key_str'~for~enumext.
4118    }
```

Messages used in length calculation.

```
4119 \msg_new:nnn { enumext } { width-negative }
4120    {
4121      Ignoring ~ negative ~ value ~ '#1=#2' ~ \msg_line_context:.\\
4122      The ~ key ~ '#1'~ accepts ~ values  ~ >= ~ 0pt.
4123    }
4124 \msg_new:nnn { enumext } { width-zero }
4125    {
4126      Invalid ~ '#1=#2' ~ \msg_line_context:.\\
4127      The ~ key ~ '#1'~ accepts ~ values  ~ > ~ 0pt.
4128    }
```

Messages used by show-length key in enumext.

```
4129 \msg_new:nnn { enumext } { list-lengths }
4130    {
4131      **** ~ Lengths ~ used ~ by ~ 'enumext' ~ level ~ '#2' ~ \msg_line_context:~\c_space_tl ****\\
```

```
4132      \__enumext_show_length:nnn { dim  } { labelsep     } {#1}
4133      \__enumext_show_length:nnn { dim  } { labelwidth   } {#1}
4134      \__enumext_show_length:nnn { dim  } { itemindent   } {#1}
4135      \__enumext_show_length:nnn { dim  } { leftmargin   } {#1}
4136      \__enumext_show_length:nnn { dim  } { rightmargin  } {#1}
4137      \__enumext_show_length:nnn { dim  } { listparindent } {#1}
4138      \__enumext_show_length:nnn { skip } { topsep       } {#1}
4139      \__enumext_show_length:nnn { skip } { parsep       } {#1}
4140      \__enumext_show_length:nnn { skip } { partopsep } {#1}
4141      \__enumext_show_length:nnn { skip } { itemsep      } {#1}
4142      ****************************************************
4143    }
```

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

```
4144 \msg_new:nnn { enumext } { list-lengths-not-nested }
4145    {
4146      **** ~ Lengths ~ used ~ by ~ '#2' ~ environment ~ \msg_line_context:~\c_space_tl ****\\
4147      \__enumext_show_length:nnn { dim  } { labelsep     } {#1}
4148      \__enumext_show_length:nnn { dim  } { labelwidth   } {#1}
4149      \__enumext_show_length:nnn { dim  } { itemindent   } {#1}
4150      \__enumext_show_length:nnn { dim  } { leftmargin   } {#1}
4151      \__enumext_show_length:nnn { dim  } { rightmargin  } {#1}
4152      \__enumext_show_length:nnn { dim  } { listparindent } {#1}
4153      \__enumext_show_length:nnn { skip } { topsep       } {#1}
4154      \__enumext_show_length:nnn { skip } { parsep       } {#1}
4155      \__enumext_show_length:nnn { skip } { partopsep } {#1}
4156      \__enumext_show_length:nnn { skip } { itemsep      } {#1}
4157      ****************************************************
4158    }
```

Messages used by ref key.

```
4159 \msg_new:nnn { enumext } { key-ref-empty }
4160    {
4161      Key ~ 'ref' ~ need ~ a ~ value ~ in ~ '#1'~ \msg_line_context:.
4162    }
```

Messages used by save-ans key.

```
4163 \msg_new:nnn { enumext } { save-ans-empty }
4164    {
4165      Key ~ 'save-ans' ~ need ~ a ~ value ~ in ~ '#1'~ \msg_line_context:.
4166    }
4167 \msg_new:nnn { enumext } { save-ans-ok }
4168    {
4169      Set ~ 'save-ans=#2' ~ in ~'#1' ~ \msg_line_context:.
4170    }
```

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

```
4171 \msg_new:nnn { enumext } { need-save-ans }
4172    {
4173      Key ~ '#1'~ works ~ only ~ with ~ the ~ 'save-ans' ~ key ~ in ~ '#2'~ \msg_line_context:.
4174    }
4175 \msg_new:nnn { enumext } { items-same-answer }
4176    {
4177      **********~Checking~answers~on~'#1'~OK~**********\\
4178      **~ All ~ items ~ stored ~ in ~ sequence ~ '#1' ~ have ~ an ~ answer. \\
4179      ****************************************
4180      \prg_replicate:nn { 7 + \str_count:n {#1} } { * }
4181    }
4182 \msg_new:nnn { enumext } { item-different-answer }
4183    {
4184      Number ~ of ~ items ~ different ~  of ~ number ~ of ~
4185      answer ~ stored ~ in ~ '#1'~ #2.
4186    }
```

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

```
4187 \msg_new:nnn { enumext } { missing-starred }
4188    {
4189      Missing ~ '\c_backslash_str #1*' ~ #2.
4190    }
4191 \msg_new:nnn { enumext } { many-starred }
4192    {
4193      Many ~ '\c_backslash_str #1*' ~ #2.
4194    }
```

Message for the nesting depth of the environment enumext.

```
4195  \msg_new:nnn { enumext } { list-too-deep }
4196    {
4197      Too ~ deep ~ nesting  ~ for ~ 'enumext' ~ \msg_line_context:.~ \\
4198      The ~ maximum  ~ level  ~ of  ~ nesting  ~ is ~ 4.
4199    }
```

Messages used by \anskey and \anspic commands.

```
4200  \msg_new:nnn { enumext } { anskey-wrong-place }
4201    {
4202      Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:.~ \\
4203      '\c_backslash_str #1' ~ works ~ in ~ the ~ environment ~ '#2'.
4204    }
4205  \msg_new:nnn { enumext } { anspic-wrong-place }
4206    {
4207      Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:.~ \\
4208      '\c_backslash_str #1' ~ works ~ in ~ the ~ environment ~ '#2'.
4209    }
4210  \msg_new:nnn { enumext } { command-wrong-place }
4211    {
4212      Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:.~ \\
4213      '\c_backslash_str #1' ~ works ~ outside ~ the ~ environment ~ '#2'.
4214    }
```

Messages used by keyans and keyanspic environment.

```
4215  \msg_new:nnn { enumext } { keyans-nested }
4216    {
4217      The ~ environment ~ 'keyans' ~ can't ~ be  ~ nested  ~ \msg_line_context:.
4218    }
4219  \msg_new:nnn { enumext } { keyans-wrong-level }
4220    {
4221      Wrong ~ level ~ position ~ for ~ 'keyans' ~ \msg_line_context:.~ \\
4222      The ~ environment ~ 'keyans' ~ can ~ only ~ be ~ in ~ the ~ first ~ level.
4223    }
4224  \msg_new:nnn { enumext } { wrong-place }
4225    {
4226      Wrong ~ place ~ for ~ '#1' ~ environment ~\msg_line_context:.~ \\
4227      '#1' ~ is ~ only ~ found ~ with ~ '#2' ~  in  ~  'enumext.
4228    }
4229  \msg_new:nnn { enumext } { keyanspic-nested }
4230    {
4231      The ~ environment ~ 'keyanspic' ~ can't ~ be  ~ nested~ \msg_line_context:.~.
4232    }
4233  \msg_new:nnn { enumext } { keyanspic-wrong-level }
4234    {
4235      Wrong ~ level ~ position ~ for ~ 'keyanspic' ~ \msg_line_context:.~ \\
4236      The ~ environment ~ 'keyans' ~ can ~ only ~ be ~ in ~ the ~ first ~ level.
4237    }
```

Messages used by \getkeyans command.

```
4238  \msg_new:nnn { enumext } { undefined-storage-anskey }
4239    {
4240      Storage ~ named ~ '#1' ~ is ~ not ~ defined ~ \msg_line_context:.
4241    }
```

Messages used by \miniright command.

```
4242  \msg_new:nnn { enumext } { missing-miniright }
4243    {
4244      Missing ~ '\c_backslash_str miniright' ~ in ~ \msg_line_context:.\\
4245      The ~ key ~ 'mini-env' ~ need ~ '\c_backslash_str miniright'.
4246    }
4247  \msg_new:nnn { enumext } { wrong-miniright-place }
4248    {
4249      Wrong ~ place ~ for ~ '\c_backslash_str miniright' ~ \msg_line_context:.~ \\
4250      Works ~ in ~ 'enumext' ~ and ~ 'keyans' ~ with ~ key ~ 'mini-env'.
4251    }
4252  \msg_new:nnn { enumext } { wrong-miniright-use }
4253    {
4254      Wrong ~ use ~ for ~ '\c_backslash_str miniright' ~ \msg_line_context:.~ \\
4255      '\c_backslash_str miniright' ~ need ~ a ~ key ~ 'mini-env'.
4256    }
```

Messages used by enumext* and keyans* environments.

```
4257 \msg_new:nnn { enumext } { nested }
4258   {
4259     The ~ starred ~ environment ~ can't ~ be ~ nested ~ \msg_line_context:.
4260   }
4261 \msg_new:nnn { enumext } { item-joined }
4262   {
4263     Items ~ joined ~ (#1) ~ > ~ #2  ~ columns ~\msg_line_context:.
4264   }
4265 \msg_new:nnn { enumext } { item-joined-columns }
4266   {
4267     Not ~ space ~ to ~ join ~ items ~ (#1) ~ > ~ #2 ~\msg_line_context:.
4268   }
```

10.41   Finish package

Finish package implementation.

```
4269 \file_input_stop:
4270 ⟨/package⟩
```

11 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.