enumext

ENUMERATE EXERCISE SHEETS

V1.0   2024-05-19[*]

©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

---

[*]This file describes a documentation for v1.0, last revised 2024-05-19.
[†]E-mail: «pablgonz@educarchile.cl».

License and Requirements

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

The enumext package loads and requires `multicol`[3] 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 `10`pt, `11`pt and `12`pt.

1 Introduction

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

4. Question with image and label below:

(A)   (B)   (C)

(D)   (E)

5. Question with image on left side:

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

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

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

* 4. (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.
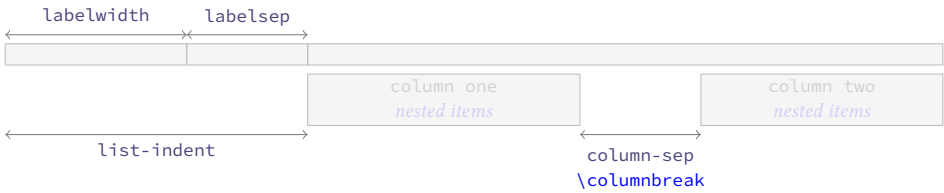

Figure 4: Representation of the two column output for a nested level in `enumext` environment.

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 the command `\centering`.

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

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

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.
* first answer

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

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

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

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

The keyans 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}
```
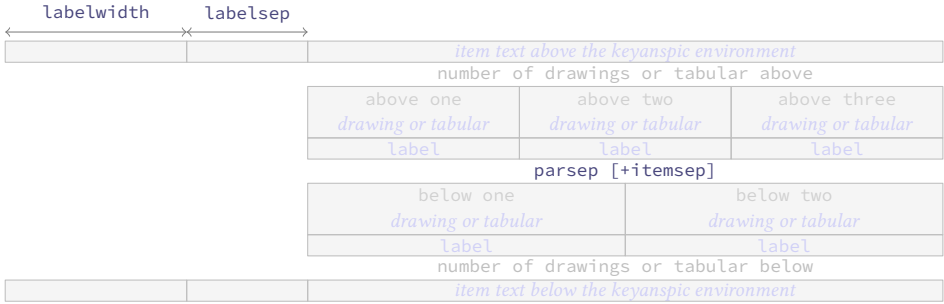
1. Question with images.

(A)

(B)

(C)

(D)

* (E)[note]
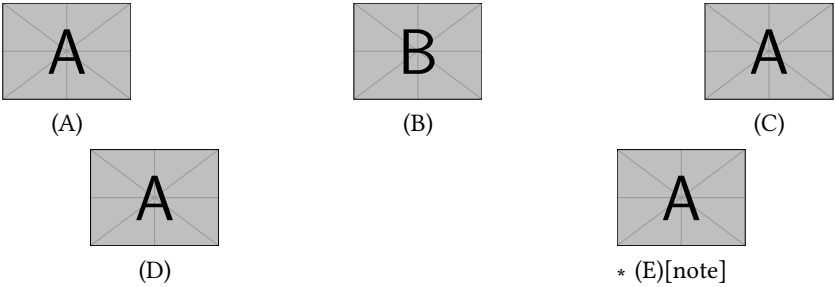
4.7 Printing stored content

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

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

   - Ⓐ value
   - Ⓑ value
   - Ⓒ value
   - Ⓓ value
   - Ⓔ value
4. Question with image and label below:

Ⓐ

Ⓑ

Ⓒ

Ⓓ

Ⓔ

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

Test keys

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

Example 4

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

🦆1 Factor $x^2 - 2x + 1$

🦆2 Factor $3x + 3y + 3z$

The following questions need to be cuaqtified :)

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

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

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

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

Example 5

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

1

Which choice best describes what happens in the passage?

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

2

Which choice best describes what happens in the passage?

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

3

Which choice best describes what happens in the passage?

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

4

Which choice best describes what happens in the passage?

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

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

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 2ε: 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 2024-05-19 – 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    {2024-05-19}
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.

\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_level_one_bool
\l__enumext_standar_bool
\g__enumext_standar_bool
\l__enumext_standar_level_one_bool
\l__enumext_keyans_env_bool

```
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_level_one_bool
28 \bool_new:N \l__enumext_standar_bool
29 \bool_new:N \g__enumext_standar_bool
30 \bool_new:N \l__enumext_standar_level_one_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 and then modified by the function __enumext_label_style:Nnn used by label key (§10.8).

```
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.*)

\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

The boolean variable \l__enumext_resume_bool is used by resume key, the value from which the environment's will start is stored in the integer variable \g__enumext_resume_int (§??). The global token list \g__enumext_item_symbol_tl is used by item-sym* key (§10.27).

```
37 \int_new:N  \g__enumext_resume_int
38 \int_new:N  \g__enumext_resume_vii_int
39 \tl_new:N   \l__enumext_resume_name_tl
40 \bool_new:N \l__enumext_resume_active_bool
41 \tl_new:N   \g__enumext_item_symbol_tl
42 \tl_new:N   \g__enumext_standar_series_tl
43 \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.

```
44 \dim_new:N \l__enumext_current_widest_dim
45 \tl_new:N  \g__enumext_counter_styles_tl
46 \tl_new:N  \g__enumext_widest_label_tl
47 \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) which determines the internal values for \leftmargin and \itemindent.

```
48 \cs_set_protected:Npn \__enumext_tmp:n #1
49   {
50     \bool_new:c { l__enumext_leftmargin_tmp_#1_bool }
51     \dim_new:c  { l__enumext_leftmargin_tmp_#1_dim }
52     \dim_new:c  { l__enumext_leftmargin_#1_dim      }
53     \dim_new:c  { l__enumext_itemindent_#1_dim      }
54   }
55 \clist_map_inline:nn { i, ii, iii, iv, v, vi, vii, viii } { \__enumext_tmp:n {#1} }
```

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

\l__enumext_multicols_above_X_skip
\l__enumext_multicols_below_X_skip

Internal variables used by columns key §10.18).

```
56 \cs_set_protected:Npn \__enumext_tmp:n #1
57   {
58     \skip_new:c  { l__enumext_multicols_above_#1_skip }
59     \skip_new:c  { l__enumext_multicols_below_#1_skip }
60   }
61 \clist_map_inline:nn { i, ii, iii, iv, v } { \__enumext_tmp:n {#1} }
```

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

`\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`

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

```
62  \int_new:N  \g__enumext_minipage_stat_int
63  \skip_new:N \l__enumext_minipage_left_skip
64  \skip_new:N \l__enumext_minipage_right_skip
65  \skip_new:N \l__enumext_minipage_after_skip
66  \skip_new:N \g__enumext_minipage_right_skip
67  \skip_new:N \g__enumext_minipage_after_skip
68  \cs_set_protected:Npn \__enumext_tmp:n #1
69    {
70      \dim_new:c  { l__enumext_minipage_left_#1_dim    }
71      \bool_new:c { l__enumext_minipage_active_#1_bool }
72    }
73  \clist_map_inline:nn { i, ii, iii, iv, v, vii, viii } { \__enumext_tmp:n {#1} }
```

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

`\l__enumext_wrap_label_X_bool`

`\l__enumext_wrap_label_opt_X_bool`

`\l__enumext_start_X_int`

`\l__enumext_fake_item_indent_X_tl`

`\l__enumext_label_fill_left_X_tl`

`\l__enumext_label_fill_right_X_tl`

`\l__enumext_vspace_a_star_X_bool`

`\l__enumext_vspace_b_star_X_bool`

The 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

```
74  \cs_set_protected:Npn \__enumext_tmp:n #1
75    {
76      \bool_new:c { l__enumext_wrap_label_#1_bool     }
77      \bool_new:c { l__enumext_wrap_label_opt_#1_bool }
78      \int_new:c  { l__enumext_start_#1_int           }
79      \tl_new:c   { l__enumext_fake_item_indent_#1_tl }
80      \tl_new:c   { l__enumext_label_fill_left_#1_tl  }
81      \tl_new:c   { l__enumext_label_fill_right_#1_tl }
82      \bool_new:c { l__enumext_vspace_a_star_#1_bool  }
83      \bool_new:c { l__enumext_vspace_b_star_#1_bool  }
84    }
85  \clist_map_inline:nn { i, ii, iii, iv, v, vii, viii } { \__enumext_tmp:n {#1} }
```

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

`\l__enumext_store_active_bool`

`\l__enumext_store_name_tl`

`\g__enumext_store_name_tl`

`\l__enumext_store_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`

`\l__enumext_keyans_tmpb_tl`

`\l__enumext_keyans_tmpa_dim`

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*` (§10.35.1) for the `keyanspic` environment.

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

```
86  \bool_new:N \l__enumext_store_active_bool
87  \tl_new:N   \l__enumext_store_name_tl
88  \tl_new:N   \g__enumext_store_name_tl
89  \tl_new:N   \l__enumext_store_anskey_arg_tl
90  \int_new:N  \l__enumext_store_columns_join_int
91  \tl_new:N   \l__enumext_store_keyans_label_tl
92  \tl_new:N   \l__enumext_store_keyans_item_opt_tl
93  \tl_new:N   \l__enumext_keyans_item_opt_tl
94  \tl_new:N   \l__enumext_keyans_tmpa_tl
95  \tl_new:N   \l__enumext_keyans_tmpb_tl
96  \dim_new:N  \l__enumext_keyans_tmpa_dim
```

(*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` (§10.40).

```
97   \tl_new:N  \l__enumext_setkey_tmpa_tl
98   \tl_new:N  \l__enumext_setkey_tmpb_tl
99   \int_new:N \l__enumext_setkey_tmpa_int
100  \seq_new:N \l__enumext_setkey_tmpa_seq
101  \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 (§10.39) and the keys columns* and columns-sep*.

```
102 \cs_set_protected:Npn \__enumext_tmp:n #1
103   {
104     \tl_new:c { l__enumext_store_opt_#1_tl              }
105     \tl_new:c { l__enumext_print_keyans_#1_tl           }
106     \bool_new:c { l__enumext_store_columns_#1_bool      }
107     \int_new:c { l__enumext_store_columns_#1_int        }
108     \bool_new:c { l__enumext_store_columns_sep_#1_bool  }
109     \dim_new:c { l__enumext_store_columns_sep_#1_dim    }
110     \bool_new:c { l__enumext_store_upper_level_#1_bool  }
111   }
112 \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).

```
113 \bool_new:N \l__enumext_show_answer_bool
114 \bool_new:N \l__enumext_show_position_bool
115 \tl_new:N    \l__enumext_mark_ref_sym_tl
116 \tl_new:N    \l__enumext_mark_answer_sym_tl
117 \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 (§10.35.2).

```
118 \seq_new:N   \l__enumext_keyans_pic_body_seq
119 \dim_new:N   \l__enumext_keyans_pic_width_dim
120 \int_new:N   \l__enumext_keyans_pic_above_int
121 \int_new:N   \l__enumext_keyans_pic_below_int
122 \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_show_bool
\g__enumext_check_ans_show_h_bool
\g__enumext_check_ans_item_tl
\g__enumext_count_item_anskey_int
\g__enumext_count_item_number_int

Internal variables used by *"check answer"* mechanism (§10.23) controlled by the check-ans and no-store keys.

```
123 \bool_new:N \l__enumext_store_ans_bool
124 \bool_new:N \l__enumext_check_ans_bool
125 \bool_new:N \g__enumext_check_ans_show_bool
126 \bool_new:N \g__enumext_check_ans_show_h_bool
127 \tl_new:N    \g__enumext_check_ans_item_tl
128 \int_new:N   \g__enumext_count_item_anskey_int
129 \int_new:N   \g__enumext_count_item_number_int
130 \int_new:N   \g__enumext_standar_star_env_int
131 \int_new:N   \g__enumext_starred_star_env_int
132 \int_new:N   \g__enumext_starred_keyans_star_env_int
133 \int_new:N   \g__enumext_standar_keyans_star_env_int
134 \int_new:N   \g__enumext_standar_keyans_pic_star_env_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.

```
135 \bool_new:N \l__enumext_hyperref_bool
136 \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.

```
137 \tl_new:N \l__enumext_newlabel_arg_one_tl
138 \tl_new:N \l__enumext_newlabel_arg_two_tl
```

```
139  \tl_new:N \l__enumext_store_write_aux_file_tl
140  \cs_set_protected:Npn \__enumext_tmp:n #1
141    {
142      \tl_new:c { l__enumext_label_copy_#1_tl }
143    }
144  \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.

```
145  \int_new:N \g__enumext_footnote_int
146  \seq_new:N \g__enumext_footnote_arg_seq
147  \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.)

\c__enumext_counter_style_tl
\l__enumext_ref_key_arg_tl
\l__enumext_ref_aux_tl
\l__enumext_the_counter_X_tl
\l__enumext_counter_style_for_ref_X_tl

Internal variables used by ref key (§10.17, §10.18).

```
148  \tl_const:Nn \c__enumext_counter_style_tl
149    { { arabic } { roman } { Roman } { alph } { Alph } }
150  \tl_new:N \l__enumext_ref_key_arg_tl
151  \tl_new:N \l__enumext_ref_aux_tl
152  \cs_set_protected:Npn \__enumext_tmp:n #1
153    {
154      \tl_new:c  { l__enumext_counter_style_for_ref_#1_tl }
155      \tl_new:c  { l__enumext_the_counter_#1_tl }
156      \tl_set:ce { l__enumext_the_counter_#1_tl } { \exp_not:c { theenumX#1 } }
157    }
158  \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.*)

\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.

```
159  \cs_set_protected:Npn \__enumext_tmp:n #1
160    {
161      \bool_new:c { l__enumext_item_starred_#1_bool    }
162      \int_new:c  { l__enumext_item_column_pos_#1_int  }
163      \int_new:c  { g__enumext_item_count_all_#1_int   }
164      \int_new:c  { l__enumext_joined_item_#1_int      }
165      \int_new:c  { l__enumext_joined_item_aux_#1_int  }
166      \int_new:c  { l__enumext_tmpa_#1_int             }
167      \box_new:c  { l__enumext_item_text_#1_box        }
168      \dim_new:c  { l__enumext_joined_width_#1_dim     }
169      \dim_new:c  { l__enumext_item_width_#1_dim       }
170      \tl_new:c   { g__enumext_item_symbol_aux_#1_tl   }
171      \str_new:c  { l__enumext_align_label_#1_str      }
172      \bool_new:c { g__enumext_minipage_active_#1_bool }
173      \tl_new:c   { g__enumext_miniright_code_#1_tl    }
174      \bool_new:c { g__enumext_minipage_center_#1_bool }
175      \dim_new:c  { g__enumext_minipage_right_#1_dim   }
176      \skip_new:c { g__enumext_minipage_right_#1_skip  }
177    }
178  \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.

```
179  \clist_const:Nn \c__enumext_all_envs_clist
180    {
181      {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv},
182      {keyans}{v}, {enumext*}{vii}, {keyans*}{viii}
183    }
```

(*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.

```
184 \cs_new_protected:Npn \__enumext_at_begin_document:n #1
185   {
186     \hook_gput_code:nnn {begindocument} {enumext} { #1 }
187   }
```

(*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.

```
188 \cs_new_protected:Npn \__enumext_after_env:nn #1 #2
189   {
190     \hook_gput_code:nnn {env/#1/after} {enumext} {#2}
191   }
```

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

`\__enumext_level:`

Function for check current level in `enumext`.

```
192 \cs_new:Nn \__enumext_level:
193   {
194     \int_to_roman:n { \l__enumext_level_int }
195   }
```

(*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?.

```
196 \prg_new_protected_conditional:Npnn \__enumext_if_is_int:n #1 { T, F, TF }
197   {
198     \regex_match:nnTF { ^[\+\-]?[\d]+$ } {#1} % $
199       { \prg_return_true: }
200       { \prg_return_false: }
201   }
```

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

`\__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.

```
202 \cs_new:Npn \__enumext_show_length:nnn #1 #2 #3
203   {
204     * ~ #2
205     \prg_replicate:nn { 14 - \str_count:n {#2} } { ~ }
206       = ~ \use:c { #1_use:c } { l__enumext_#2_#3_#1 } \\
207   }
```

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

`\__enumext_zero_count_level:`

Internal function used by `check-ans` key.

```
208 \cs_set_protected:Nn \__enumext_zero_count_level:
209   {
210     \cs_set_protected:Npn \__enumext_tmp:n ##1
211       {
212         \int_gzero:c { g__enumext_count_level_##1_int }
213       }
214     \clist_map_inline:nn { i, ii, iii, iv, vii } { \__enumext_tmp:n {##1} }
215   }
```

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

`\__enumext_current_env_set_bool:`

The function `\__enumext_current_env_set_bool:` will set the global variables `\g__enumext_-standar_bool` and `\g__enumext_starred_bool` with which we will distinguish whether the environments `enumext` and `enumext*` are nested in each other. This function is passed to the `\__enumext_-safe_exec:` function in the definition of the `enumext` environment (pag 76) and to the `\__enumext_-safe_exec_vii:` function in the definition of the `enumext*` environment (pag 89).

```
216 \cs_new_protected:Nn \__enumext_current_env_set_bool:
217   {
```

```
218    \str_case:en { \@currenvir }
219      {
220        {enumext}
221          {
222            \bool_lazy_and:nnT
223              { \bool_not_p:n { \g__enumext_standar_bool } }
224              { \int_compare_p:nNn { \l__enumext_level_h_int } = { \c_zero_int } }
225              {
226                \bool_gset_true:N \g__enumext_standar_bool
227                \int_gset:Nn \g__enumext_standar_star_env_int { \inputlineno }
228                \typeout{working-on-enumext}
229              }
230          }
231        {enumext*}
232          {
233            \bool_lazy_and:nnT
234              { \bool_not_p:n { \g__enumext_starred_bool } }
235              { \int_compare_p:nNn { \l__enumext_level_int } = { \c_zero_int } }
236              {
237                \bool_gset_true:N \g__enumext_starred_bool
238                \int_gset:Nn \g__enumext_starred_star_env_int { \inputlineno }
239                \typeout{working-on-enumext*}
240              }
241          }
242      }
243  }
```

(*End of definition for* __enumext_current_env_set_bool:.)

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.

```
244  \__enumext_at_begin_document:n
245    {
246      \cs_new_eq:NN \__enumext_start_list:nn \list
247      \cs_new_eq:NN \__enumext_stop_list: \endlist
248      \cs_new_eq:NN \__enumext_item_std:w \item
249    }
```

(*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.

```
250  \__enumext_at_begin_document:n
251    {
252      \cs_new_eq:NN \__enumext_minipage:w \minipage
253      \cs_new_eq:NN \__enumext_endminipage: \endminipage
254    }
```

(*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.

```
255  \hook_gput_code:nnn { begindocument } { enumext } { \__enumext_after_hyperref: }
256  \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".

```
257  \cs_new_protected:Nn \__enumext_after_hyperref:
258    {
259      \IfPackageLoadedTF { hyperref }
260        {
261          \msg_info:nnn { enumext } { package-load } { hyperref }
262          \bool_set_true:N \l__enumext_hyperref_bool
263          \IfHyperBoolean{hyperfootnotes}
264            {
265              \typeout{hyperfootnotes=true}
266              \bool_set_true:N \l__enumext_footnotes_key_bool
267            }
268            { \typeout{hyperfootnotes=false} }
269        }
270        {  }
```

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.

```
271      \bool_if:NT \l__enumext_footnotes_key_bool
272        {
273          \IfPackageLoadedTF { footnotehyper }
274            {
275              \msg_info:nnn { enumext } { package-load } { footnotehyper }
276            }
277            {
278              \typeout{No ~ footnotehyper ~ load}
279              \typeout{Load ~ and  ~ use  ~ \string\makesavenoteenv{enumext*}}
280              \bool_set_false:N \l__enumext_footnotes_key_bool
281            }
282        }
```

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.

```
283      \bool_if:NTF \l__enumext_hyperref_bool
284        {
285          \cs_new_eq:NN \__enumext_hypertarget:nn \hypertarget
286          \cs_new_eq:NN \__enumext_phantomsection: \phantomsection
287        }
288        {
289          \cs_new_eq:NN \__enumext_hypertarget:nn \use_none:nn
290          \cs_new_eq:NN \__enumext_phantomsection: \prg_do_nothing:
291        }
292    }
```

(*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.

```
293  \cs_new_protected:Npn \__enumext_newlabel:nn #1 #2
294    {
295      \protected@write \@auxout { }
296        {
297          \token_to_str:N \newlabel {#1}
```

```
298            {
299              {#2}
300              \bool_if:NT \l__enumext_hyperref_bool
301                { { \thepage } {#2} {#1} }
302                { }
303            }
304          }
305      \__enumext_hypertarget:nn {#1} { }
306      \__enumext_phantomsection:
307    }
```

(*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.

```
308 \cs_new_protected:Npn \__enumext_define_counters:Nn #1 #2
309   {
310     \cs_if_exist:cTF { c@ #2 }
311       { \msg_fatal:nnn { enumext } { counters }{ #2 } }
312       {
313         \tl_set:Nn #1 { #2 }
314         \newcounter { #2 }
315       }
316   }
```

(*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.

```
317 \__enumext_define_counters:Nn \l__enumext_counter_i_tl    { enumXi     }
318 \__enumext_define_counters:Nn \l__enumext_counter_ii_tl   { enumXii    }
319 \__enumext_define_counters:Nn \l__enumext_counter_iii_tl  { enumXiii   }
320 \__enumext_define_counters:Nn \l__enumext_counter_iv_tl   { enumXiv    }
321 \__enumext_define_counters:Nn \l__enumext_counter_v_tl    { enumXv     }
322 \__enumext_define_counters:Nn \l__enumext_counter_vi_tl   { enumXvi    }
323 \__enumext_define_counters:Nn \l__enumext_counter_vii_tl  { enumXvii   }
324 \__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.

```
325 \cs_new_protected:Npn \__enumext_register_counter_style:Nn #1 #2
326   {
327     \tl_const:cn { c__enumext_widest_ \cs_to_str:N #1 _tl } {#2}
328     \tl_gput_right:Nn \g__enumext_counter_styles_tl {#1}
329   }
330 \__enumext_register_counter_style:Nn \arabic { 0 }
331 \__enumext_register_counter_style:Nn \Alph   { M }
332 \__enumext_register_counter_style:Nn \alph   { m }
333 \__enumext_register_counter_style:Nn \Roman  { VIII }
334 \__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.

```
335  \cs_new_protected:Npn \__enumext_label_width_by_box:Nn #1 #2
336    {
337      \hbox_set:Nn \l__enumext_label_width_by_box {#2}
338      \dim_set:Nn #1 { \box_wd:N \l__enumext_label_width_by_box }
339    }
340  \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.

```
341  \cs_new_protected:Npn \__enumext_label_style:Nnn #1 #2 #3
342    {
343      \tl_clear_new:N #1
344      \tl_put_right:Ne #1 { \tl_trim_spaces:n {#3} }
345      \tl_gset_eq:NN \g__enumext_widest_label_tl #1
346      \tl_map_inline:Nn \g__enumext_counter_styles_tl
347        {
348          \tl_replace_all:Nne #1 { ##1* } { \exp_not:N ##1 {#2} }
349          \tl_greplace_all:Nne \g__enumext_widest_label_tl { ##1* }
350            { \tl_use:c { c__enumext_widest_ \cs_to_str:N ##1 _tl } }
351        }
352      \__enumext_label_width_by_box:Nn \l__enumext_current_widest_dim
353        { \tl_use:N \g__enumext_widest_label_tl }
354      \tl_set_eq:cN { the #2 } #1
355    }
356  \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.

```
357  \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
358    {
359      \keys_define:nn { enumext / #1 }
360        {
361          font       .tl_set:c   = { l__enumext_label_font_style_#2_tl },
362          font       .value_required:n = true,
363          labelsep   .dim_set:c  = { l__enumext_labelsep_#2_dim },
364          labelsep   .initial:n  = {0.3333em},
365          labelsep   .value_required:n = true,
366          labelwidth .dim_set:c  = { l__enumext_labelwidth_#2_dim },
367          labelwidth .value_required:n = true,
368          wrap-label  .cs_set_protected:cp = { __enumext_wrapper_label_#2:n } ##1,
369          wrap-label  .initial:n  = {##1},
370          wrap-label  .value_required:n = true,
371          wrap-label* .code:n = {
372                              \bool_set_true:c { l__enumext_wrap_label_opt_#2_bool }
373                              \keys_set:nn { enumext / #1 } { wrap-label = {##1} }
374                            },
375          wrap-label* .value_required:n = true,
376        }
377    }
378  \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.

```
379  \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
380    {
```

```
381    \keys_define:nn { enumext / #1 }
382      {
383        align .choice:,
384        align / left   .code:n =
385                        {
386                          \tl_clear:c { l__enumext_label_fill_left_#2_tl  }
387                          \tl_set:cn  { l__enumext_label_fill_right_#2_tl } { \hfill }
388                        },
389        align / right  .code:n =
390                        {
391                          \tl_set:cn  { l__enumext_label_fill_left_#2_tl  } { \hfill }
392                          \tl_clear:c { l__enumext_label_fill_right_#2_tl }
393                        },
394        align / center .code:n =
395                        {
396                          \tl_set:cn { l__enumext_label_fill_left_#2_tl  } { \hfill }
397                          \tl_set:cn { l__enumext_label_fill_right_#2_tl } { \hfill }
398                        },
399        align .initial:n  = left,
400        align .value_required:n  = true,
401      }
402    }
403  \clist_map_inline:nn
404    {
405      {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {keyans}{v}
406    }
407    { \__enumext_tmp:nn #1 }
```

Definition of `align` key for `enumext*` and `keyans*` environments.

```
408  \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
409    {
410      \keys_define:nn { enumext / #1 }
411        {
412          align .choice:,
413          align / left   .code:n = \str_set:cn { l__enumext_align_label_#2_str } { l },
414          align / right  .code:n = \str_set:cn { l__enumext_align_label_#2_str } { r },
415          align / center .code:n = \str_set:cn { l__enumext_align_label_#2_str } { c },
416          align .initial:n = left,
417          align .value_required:n = true,
418        }
419    }
420  \clist_map_inline:nn { {enumext*}{vii}, {keyans*}{viii} } { \__enumext_tmp:nn #1 }
```

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

10.11   Setting `label` and `ref` keys

`\__enumext_regex_label_ref_key:` The internal function `\__enumext_regex_label_ref_key:` replace the * with the actual counter of the running level and is used by the `\__enumext_set_label_ref:n` function.

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.

```
421  \cs_new_protected:Nn \__enumext_regex_label_ref_key:
422    {
423      \tl_map_inline:Nn \c__enumext_counter_style_tl
424        {
425          \regex_replace_once:nnN { \c{##1}\* }
426            { \c{##1}\cB{\u{l__enumext_ref_aux_tl}\cE} } \l__enumext_ref_key_arg_tl
427        }
428    }
```

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

`\__enumext_set_label_ref:n` The `\__enumext_set_label_ref:n` function controlled by the `ref` key is in charge of handling the customization of the reference system.

First we will set the variable `\l__enumext_the_counter_X_tl` according to the command created for *each counter*, apply the *regex* function `\__enumext_regex_label_ref_key:` and then renew the command and save it in the variable `\l__enumext_counter_style_for_ref_X_tl`.

```
429  \cs_new_protected:Npn \__enumext_set_label_ref:n #1
430    {
431      \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
```

```
432    \tl_set_eq:Nc \l__enumext_ref_aux_tl { l__enumext_counter_ \__enumext_level: _tl }
433    \__enumext_regex_label_ref_key:
434    \tl_set_eq:Nc \l__enumext_ref_aux_tl { l__enumext_the_counter_ \__enumext_level: _tl }
435    \tl_put_right:ce { l__enumext_counter_style_for_ref_ \__enumext_level: _tl }
436      {
437        \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_aux_tl }
438          { \exp_not:V \l__enumext_ref_key_arg_tl }
439      }
440    }
```

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

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

```
441  \cs_new_protected:Nn \__enumext_use_key_ref:
442    {
443      \tl_if_empty:cF { l__enumext_counter_style_for_ref_ \__enumext_level: _tl }
444        {
445          \tl_use:c { l__enumext_counter_style_for_ref_ \__enumext_level: _tl }
446        }
447    }
```

(*End of definition for* __enumext_use_key_ref:.)

For enumext* and keyans* environments the situation is a bit different since hyperref interferes here (I am not clear why), so we will define a new function to execute the task.

To handle that we will look at the nesting level of the starred environments, later I will run the constraint functions to make everything OK.

__enumext_set_label_ref_h:n   The __enumext_set_label_ref_h:n function controlled by the ref key is in charge of handling the customization of the reference system.

First we will set the variable \l__enumext_the_counter_X_tl according to the command created for *each counter*, apply the *regex* function __enumext_regex_label_ref_key: and then renew the command and save it in the variable \l__enumext_counter_style_for_ref_X_tl.

```
448  \cs_new_protected:Npn \__enumext_set_label_ref_h:n #1
449    {
450      \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
451      \int_compare:nNnTF { \l__enumext_level_h_int } = { 1 }
452        {
453          \tl_set_eq:NN \l__enumext_ref_aux_tl \l__enumext_counter_vii_tl
454          \__enumext_regex_label_ref_key:
455          \tl_set_eq:NN \l__enumext_ref_aux_tl \l__enumext_the_counter_vii_tl
456          \tl_put_right:Ne \l__enumext_counter_style_for_ref_vii_tl
457            {
458              \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_aux_tl }
459                { \exp_not:V \l__enumext_ref_key_arg_tl }
460            }
461        }
462        {
463          \tl_set_eq:NN \l__enumext_ref_aux_tl \l__enumext_counter_viii_tl
464          \__enumext_regex_label_ref_key:
465          \tl_set_eq:NN \l__enumext_ref_aux_tl \l__enumext_the_counter_viii_tl
466          \tl_put_right:Ne \l__enumext_counter_style_for_ref_vii_tl
467            {
468              \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_aux_tl }
469                { \exp_not:V \l__enumext_ref_key_arg_tl }
470            }
471        }
472    }
```

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

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

```
473  \cs_new_protected:Nn \__enumext_use_key_ref_h:
474    {
475      \int_compare:nNnTF { \l__enumext_level_h_int } = { 1 }
476        {
477          \tl_if_empty:NF \l__enumext_counter_style_for_ref_vii_tl
478            {
479              \tl_use:N \l__enumext_counter_style_for_ref_vii_tl
```

```
480                    }
481                }
482            {
483                \tl_if_empty:NF \l__enumext_counter_style_for_ref_viii_tl
484                    {
485                      \tl_use:N \l__enumext_counter_style_for_ref_viii_tl
486                    }
487            }
488        }
```

(*End of definition for* __enumext_use_key_ref_h:.)

10.11.1   Define and set `label` key 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.

```
489  \cs_set_protected:Npn \__enumext_tmp:nnn #1 #2 #3
490    {
491      \keys_define:nn { enumext / #1 }
492        {
493          label .code:n    = {
494                              \__enumext_label_style:cvn { l__enumext_label_#2_tl }
495                              { l__enumext_counter_#2_tl } {##1}
496                              \dim_set_eq:cN  { l__enumext_labelwidth_#2_dim }
497                              \l__enumext_current_widest_dim
498                            },
499          label .initial:n = #3,
500          label .value_required:n = true,
501          ref   .code:n    = \__enumext_set_label_ref:n {##1},
502          ref   .value_required:n = true,
503        }
504    }
505  \__enumext_tmp:nnn { level-1 } {   i } { \arabic*.}
506  \__enumext_tmp:nnn { level-2 } {  ii } { (\alph*) }
507  \__enumext_tmp:nnn { level-3 } { iii } { \roman*. }
508  \__enumext_tmp:nnn { level-4 } {  iv } { \Alph*.  }
```

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

10.11.2   Define and set `label` key 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.

```
509  \cs_set_protected:Npn \__enumext_tmp:nnn #1 #2 #3
510    {
511      \keys_define:nn { enumext / #1 }
512        {
513          label .code:n    = {
514                              \__enumext_label_style:cvn { l__enumext_label_#2_tl }
515                              { l__enumext_counter_#2_tl } {##1}
516                              \dim_set_eq:cN  { l__enumext_labelwidth_#2_dim }
517                              \l__enumext_current_widest_dim
518                            },
519          label .initial:n = #3,
520          label .value_required:n = true,
521          ref   .code:n    = \__enumext_set_label_ref_h:n {##1},
522          ref   .value_required:n = true,
523        }
524    }
525  \__enumext_tmp:nnn { enumext* } {  vii } { \arabic*.}
526  \__enumext_tmp:nnn { keyans*  } { viii } { (\Alph*) }
```

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

10.11.3   Define and set `label` key for `keyans` and `keyanspic` environment

label

\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. The keyanspic environment use the same ⟨*label*⟩ as the keyans environment. Define and set label key for keyans environment.

```
527  \keys_define:nn { enumext / keyans }
528    {
529      label .code:n    = {
530                          \__enumext_label_style:cvn { l__enumext_label_v_tl }
531                          { l__enumext_counter_v_tl } {#1}
```

```
532                              \dim_set_eq:cN  { l__enumext_labelwidth_v_dim }
533                                \l__enumext_current_widest_dim
534                              \__enumext_label_style:cvn { l__enumext_label_vi_tl }
535                                { l__enumext_counter_vi_tl } {#1}
536                              \dim_set_eq:cN  { l__enumext_labelwidth_v_dim }
537                                \l__enumext_current_widest_dim
538                          },
539        label .initial:n = (\Alph*),
540        label .value_required:n = true,
541      }
```

(*End of definition for* label *,* \l__enumext_label_v_tl *, and* \l__enumext_label_vi_tl*.*)

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.

```
542  \cs_new_protected:Npn \__enumext_start_from:NNn #1 #2 #3
543    {
544      \__enumext_if_is_int:nTF { #3 }
545        {
546          \int_set:Nn #2 {#3}
547        }
548        {
549          \regex_match:nVT { \c{Alph} | \c{alph} } {#1}
550            { \int_set:Nn #2 { \int_from_alph:n {#3} } }
551          \regex_match:nVT { \c{Roman} | \c{roman} } {#1}
552            { \int_set:Nn #2  { \int_from_roman:n {#3} } }
553        }
554    }
555  \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.

```
556  \cs_new_protected:Npn \__enumext_widest_from:nNNn #1 #2 #3 #4
557    {
558      \__enumext_if_is_int:nTF {#4}
559        {
560          \setcounter{enumX#1} { #4 }
561        }
562        {
563          \regex_match:nVT { \c{Alph} | \c{alph} } {#2}
564            { \setcounter{enumX#1} { \int_from_alph:n {#4} } }
565          \regex_match:nVT { \c{Roman} | \c{roman} } {#2}
566            { \setcounter{enumX#1} { \int_from_roman:n {#4} } }
567        }
568      \__enumext_label_width_by_box:cv
569        { l__enumext_labelwidth_#1_dim } { l__enumext_label_#1_tl }
570    }
571  \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.

```
572  \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
573    {
```

```
574      \keys_define:nn { enumext / #1 }
575        {
576          start  .code:n   = {
577                                \__enumext_start_from:ccn
578                                  { l__enumext_label_#2_tl }
579                                  { l__enumext_start_#2_int } {##1}
580                              },
581          start  .initial:n = 1,
582          widest .code:n   = {
583                                \__enumext_widest_from:nccn {#2}
584                                  { l__enumext_label_#2_tl }
585                                  { l__enumext_labelwidth_#2_dim } {##1}
586                              },
587          widest .value_required:n = true,
588          start  .value_required:n = true,
589        }
590    }
591 \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.

```
592 \cs_set_protected:Npn \__enumext_tmp:nnnnnn #1 #2 #3 #4 #5 #6
593    {
594      \keys_define:nn { enumext / #1 }
595        {
596          topsep    .skip_set:c = { l__enumext_topsep_#2_skip },
597          topsep    .initial:n  = {#3},
598          topsep    .value_required:n = true,
599          partopsep .skip_set:c = { l__enumext_partopsep_#2_skip },
600          partopsep .initial:n  = {#4},
601          partopsep .value_required:n = true,
602          parsep    .skip_set:c = { l__enumext_parsep_#2_skip },
603          parsep    .initial:n  = {#5},
604          parsep    .value_required:n = true,
605          itemsep   .skip_set:c = { l__enumext_itemsep_#2_skip },
606          itemsep   .initial:n  = {#6},
607          itemsep   .value_required:n = true,
608          noitemsep .meta:n     = { itemsep = 0pt, parsep = 0pt },
609          noitemsep .value_forbidden:n = true,
610          nosep     .meta:n     = {
611                                    itemsep = 0pt, parsep= 0pt,
612                                    topsep = 0pt, partopsep = 0pt,
613                                  },
614          nosep     .value_forbidden:n = true,
615        }
616    }
```

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

```
617 \__enumext_tmp:nnnnnn { level-1 } { i } { 8.0pt plus 2.0pt minus 4.0pt }
618    { 2.0pt plus 1.0pt minus 1.0pt } { 4.0pt plus 2.0pt minus 1.0pt }
619    { 4.0pt plus 2.0pt minus 1.0pt }
620 \__enumext_tmp:nnnnnn { level-2 } { ii } { 4.0pt plus 2.0pt minus 1.0pt }
621    { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
622    { 2.0pt plus 1.0pt minus 1.0pt }
623 \__enumext_tmp:nnnnnn { level-3 } { iii } { 2.0pt plus 1.0pt minus 1.0pt }
624    { 1.0pt minus 1.0pt }{ 0pt }{ 2.0pt plus 1.0pt minus 1.0pt }
625 \__enumext_tmp:nnnnnn { level-4 } { iv } { 2.0pt plus 1.0pt minus 1.0pt }
626    { 1.0pt minus 1.0pt }{ 0pt }{ 2.0pt plus 1.0pt minus 1.0pt }
627 \__enumext_tmp:nnnnnn { keyans  } { v }{ 4.0pt plus 2.0pt minus 1.0pt }
628    { 2.0pt plus 1.0pt minus 1.0pt }{ 2.0pt plus 1.0pt minus 1.0pt }
629    { 2.0pt plus 1.0pt minus 1.0pt }
630 \__enumext_tmp:nnnnnn { enumext* } { vii } { 8.0pt plus 2.0pt minus 4.0pt }
631    { 2.0pt plus 1.0pt minus 1.0pt } { 4.0pt plus 2.0pt minus 1.0pt }
632    { 4.0pt plus 2.0pt minus 1.0pt }
633 \__enumext_tmp:nnnnnn { keyans* } { viii } { 4.0pt plus 2.0pt minus 1.0pt }
634    { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
635    { 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.

```
636 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
637   {
638     \keys_define:nn { enumext / #1 }
639       {
640         itemindent    .dim_set:c = { l__enumext_fake_item_indent_#2_dim },
641         itemindent    .value_required:n = true,
642         rightmargin   .dim_set:c = { l__enumext_rightmargin_#2_dim },
643         rightmargin   .value_required:n = true,
644         listparindent .dim_set:c = { l__enumext_listparindent_#2_dim },
645         listparindent .value_required:n = true,
646         list-offset   .dim_set:c = { l__enumext_listoffset_#2_dim },
647         list-offset   .value_required:n = true,
648         list-indent   .code:n    =
649                           \bool_set_true:c { l__enumext_leftmargin_tmp_#2_bool }
650                           \dim_set:cn { l__enumext_leftmargin_tmp_#2_dim } {##1},
651         list-indent   .value_required:n = true,
652       }
653   }
654 \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.

```
655 \cs_set_protected:Npn \__enumext_tmp:n #1
656   {
657     \keys_define:nn { enumext / #1 } { list-indent .initial:n  = 0pt, }
658   }
659 \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.

```
660 \cs_set_protected:Nn \__enumext_fake_item:
661   {
662     \dim_compare:nNnT
663       { \dim_use:c { l__enumext_fake_item_indent_ \__enumext_level: _dim } }
664       >
665       { \c_zero_dim }
666       {
667         \tl_set:ce { l__enumext_fake_item_indent_ \__enumext_level: _tl }
668           {
669             \exp_not:N \mode_leave_vertical:
670             \exp_not:n { \skip_horizontal:n }
671               { \dim_use:c { l__enumext_fake_item_indent_ \__enumext_level: _dim } }
672             \ignorespaces
673           }
674       }
675   }
676 \cs_set_protected:Nn \__enumext_keyans_fake_item:
677   {
678     \dim_compare:nNnT
679       { \l__enumext_fake_item_indent_v_dim } > { \c_zero_dim }
680       {
681         \tl_set:Ne \l__enumext_fake_item_indent_v_tl
682           {
683             \exp_not:N \mode_leave_vertical:
684             \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_v_dim
685           }
686       }
687   }
688 \cs_set_protected:Nn \__enumext_fake_item_vii:
689   {
690     \dim_compare:nNnT
691       { \l__enumext_fake_item_indent_vii_dim } > { \c_zero_dim }
```

```
692        {
693            \tl_set:Ne \l__enumext_fake_item_indent_vii_tl
694              {
695                \exp_not:N \mode_leave_vertical:
696                \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_vii_dim
697              }
698          }
699      }
700  \cs_set_protected:Nn \__enumext_fake_item_viii:
701    {
702      \dim_compare:nNnT
703        { \l__enumext_fake_item_indent_viii_dim } > { \c_zero_dim }
704        {
705          \tl_set:Ne \l__enumext_fake_item_indent_viii_tl
706            {
707              \exp_not:N \mode_leave_vertical:
708              \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_viii_dim
709            }
710        }
711    }
```

(*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.

```
712  \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
713    {
714      \keys_define:nn { enumext / #1 }
715        {
716          show-length .bool_set:c = { l__enumext_show_length_#2_bool },
717          show-length .initial:n  = false,
718        }
719    }
720  \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 Define and set `before`, `before*`, `after` and `first` keys for `enumext` and `keyans` environments.
before*
after
first
```
721  \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
722    {
723      \keys_define:nn { enumext / #1 }
724        {
725          before  .tl_set:c   = { l__enumext_before_no_starred_key_#2_tl },
726          before  .value_required:n = true,
727          before* .tl_set:c   = { l__enumext_before_starred_key_#2_tl },
728          before* .value_required:n = true,
729          after   .tl_set:c   = { l__enumext_after_stop_list_#2_tl },
730          after   .value_required:n = true,
731          first   .tl_set:c   = { l__enumext_after_list_args_#2_tl },
732          first   .value_required:n = true,
733        }
734    }
735  \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: The function `\__enumext_before_args_exec:` executes the {⟨*code*⟩} set by the `before*` key *"before"*
__enumext_before_keys_exec: the `enumext` environment is started. The {⟨*code*⟩} is executed *"without"* knowing any definition of the
__enumext_after_stop_list: *second argument* of the list.
__enumext_after_args_exec:
```
736  \cs_new_protected:Nn \__enumext_before_args_exec:
737    {
738      \tl_use:c { l__enumext_before_starred_key_ \__enumext_level: _tl }
739    }
```

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*⟩.

```
740  \cs_new_protected:Nn \__enumext_before_keys_exec:
741    {
742      \tl_use:c { l__enumext_before_no_starred_key_ \__enumext_level: _tl }
743    }
```

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

```
744  \cs_new_protected:Nn \__enumext_after_stop_list:
745    {
746      \tl_use:c { l__enumext_after_stop_list_ \__enumext_level: _tl }
747    }
```

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.

```
748  \cs_new_protected:Nn \__enumext_after_args_exec:
749    {
750      \tl_use:c { l__enumext_after_list_args_ \__enumext_level: _tl }
751    }
```

(*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.

```
752  \cs_new_protected:Nn \__enumext_before_args_exec_v:
753    {
754      \tl_use:N \l__enumext_before_starred_key_v_tl
755    }
```

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*⟩.

```
756  \cs_new_protected:Nn \__enumext_before_keys_exec_v:
757    {
758      \tl_use:N \l__enumext_before_no_starred_key_v_tl
759    }
```

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

```
760  \cs_new_protected:Nn \__enumext_after_stop_list_v:
761    {
762      \tl_use:N \l__enumext_after_stop_list_v_tl
763    }
```

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.

```
764  \cs_new_protected:Nn \__enumext_after_args_exec_v:
765    {
766      \tl_use:N \l__enumext_after_list_args_v_tl
767    }
```

(*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.

```
768  \cs_new_protected:Nn \__enumext_before_args_exec_vii:
769    {
770      \tl_use:N \l__enumext_before_starred_key_vii_tl
771    }
772  \cs_new_protected:Nn \__enumext_before_args_exec_viii:
773    {
774      \tl_use:N \l__enumext_before_starred_key_viii_tl
775    }
```

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*⟩.

```
776 \cs_new_protected:Nn \__enumext_before_keys_exec_vii:
777   {
778     \tl_use:N \l__enumext_before_no_starred_key_vii_tl
779   }
780 \cs_new_protected:Nn \__enumext_before_keys_exec_viii:
781   {
782     \tl_use:N \l__enumext_before_no_starred_key_viii_tl
783   }
```

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

```
784 \cs_new_protected:Nn \__enumext_after_stop_list_vii:
785   {
786     \tl_use:N \l__enumext_after_stop_list_vii_tl
787   }
788 \cs_new_protected:Nn \__enumext_after_stop_list_viii:
789   {
790     \tl_use:N \l__enumext_after_stop_list_viii_tl
791   }
```

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`.

```
792 \cs_new_protected:Nn \__enumext_after_args_exec_vii:
793   {
794     \tl_use:N \l__enumext_after_list_args_vii_tl
795   }
796 \cs_new_protected:Nn \__enumext_after_args_exec_viii:
797   {
798     \tl_use:N \l__enumext_after_list_args_viii_tl
799   }
```

(*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.

```
800 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
801   {
802     \keys_define:nn { enumext / #1 }
803       {
804         mini-env    .dim_set:c  = { l__enumext_minipage_right_#2_dim },
805         mini-env    .value_required:n = true,
806         mini-sep    .dim_set:c  = { l__enumext_minipage_hsep_#2_dim },
807         mini-sep    .initial:n  = 0.3333em,
808         mini-sep    .value_required:n = true,
809         columns-sep .dim_set:c  = { l__enumext_columns_sep_#2_dim },
810         columns-sep .value_required:n = true,
811         columns     .int_set:c  = { l__enumext_columns_#2_int },
812         columns     .initial:n  = 1,
813         columns     .value_required:n = true,
814       }
815   }
816 \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`.

```
817 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
818   {
819     \keys_define:nn { enumext / #1 }
820       {
821         columns    .initial:n = 2,
822         miniright  .tl_gset:c = { g__enumext_miniright_code_#2_tl },
823         miniright  .value_required:n = true,
824         miniright* .code:n    = {
```

```
825                              \bool_gset_true:c { g__enumext_minipage_center_#2_bool }
826                              \keys_set:nn { enumext / #1 } { miniright = {##1} }
827                           },
828          miniright* .value_required:n = true,
829       }
830    }
831  \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.
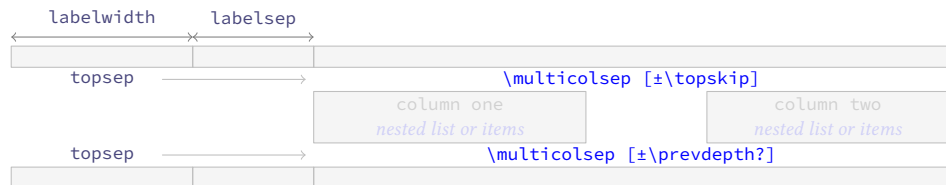
Figure 7: Representation of the vertical space in multicols for a nested level.

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*.

```
832  \cs_new_protected:Nn \__enumext_multi_set_vskip:
833    {
834      \skip_set:cn { l__enumext_multicols_above_ \__enumext_level: _skip }
835        {
836          \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
837        }
838      \skip_set:cn { l__enumext_multicols_below_ \__enumext_level: _skip }
839        {
840          \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
841        }
842      \__enumext_add_pre_parsep:
843    }
```

(*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*.

```
844  \cs_new_protected:Nn \__enumext_add_pre_parsep:
845    {
846      \int_case:nn { \l__enumext_level_int }
847        {
848          { 2 }{
849                \skip_if_eq:nnF { \l__enumext_parsep_i_skip } { \c_zero_skip }
850                  {
851                    \skip_add:Nn \l__enumext_multicols_above_ii_skip { \l__enumext_parsep_i_skip }
852                  }
853              }
854          { 3 }{
855                \skip_if_eq:nnF { \l__enumext_parsep_ii_skip } { \c_zero_skip }
856                  {
```

```
857                        \skip_add:Nn \l__enumext_multicols_above_iii_skip { \l__enumext_parsep_ii_skip
858                    }
859            }
860        { 4 }{
861                \skip_if_eq:nnF { \l__enumext_parsep_iii_skip } { \c_zero_skip }
862                {
863                \skip_add:Nn \l__enumext_multicols_above_iv_skip { \l__enumext_parsep_iii_skip
864                }
865            }
866        }
867    }
```

(*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*⟩.

```
868  \cs_new_protected:Nn \__enumext_multi_addvspace:
869    {
870      \__enumext_multi_set_vskip:
871      \mode_if_vertical:T
872        {
873          \skip_add:cn { l__enumext_multicols_above_ \__enumext_level: _skip }
874            {
875              \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
876            }
877          \skip_add:cn { l__enumext_multicols_below_ \__enumext_level: _skip }
878            {
879              \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
880            }
881        }
882      \par\nopagebreak
883      \addvspace{ \skip_use:c { l__enumext_multicols_above_ \__enumext_level: _skip } }
884    }
```

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

10.18.2 Adjustment of vertical spaces for multicols in keyans

__enumext_keyans_multi_set_vskip: The function __enumext_keyans_multi_set_vskip: will take care of determining the *"adjusted*
__enumext_keyans_multi_addvspace: *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.

```
885  \cs_new_protected:Nn \__enumext_keyans_multi_set_vskip:
886    {
887      \skip_set:Nn \l__enumext_multicols_above_v_skip
888        {
889          \l__enumext_topsep_v_skip
890        }
891      \skip_set:Nn \l__enumext_multicols_below_v_skip
892        {
893          \l__enumext_topsep_v_skip
894        }
895    }
896  \cs_new_protected:Nn \__enumext_keyans_multi_addvspace:
897    {
898      \__enumext_keyans_multi_set_vskip:
899      \mode_if_vertical:T
900        {
901          \skip_add:Nn \l__enumext_multicols_above_v_skip
902            {
903              \skip_use:N \l__enumext_partopsep_v_skip
904            }
905          \skip_add:Nn \l__enumext_multicols_below_v_skip
906            {
907              \skip_use:N \l__enumext_partopsep_v_skip
908            }
909        }
910      \par\nopagebreak
911      \addvspace{ \l__enumext_multicols_above_v_skip }
912    }
```

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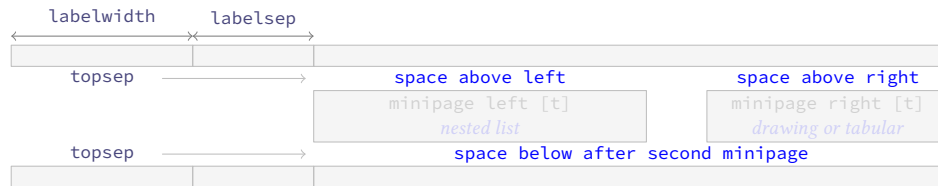
Figure 8: Representation of the `minipage` spacing adjustment for a nested level.

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.

```
913  \DeclareDocumentEnvironment{__enumext_mini_env*}{ m }
914    {
915      \__enumext_minipage:w [ t ] { #1 }
916        \legacy_if_gset_false:n { @minipage }
917        \vspace { 0pt }
918    }
919    { \__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`.

```
920  \cs_new_protected:Nn \__enumext_mini_set_vskip:
921    {
922      \int_compare:nNnTF
923        { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } > { 1 }
924        {
```

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.

```
925          \skip_if_eq:nnTF
926            { \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip } } { \c_zero_skip }
927            {
928              \skip_set:Nn \l__enumext_minipage_left_skip
929                {
930                  -0.150\box_dp:N \strutbox
931                }
932              \skip_set:Nn \l__enumext_minipage_right_skip
933                {
934                  0.695\box_dp:N \strutbox
935                }
```

```
936              \skip_set:Nn \l__enumext_minipage_after_skip
937                  {
938                      \box_dp:N \strutbox
939                  }
940              \__enumext_zero_parsep:
941          }
942          {
943              \skip_set:Nn \l__enumext_minipage_left_skip
944                  {
945                      \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
946                  }
947              \skip_set:Nn \l__enumext_minipage_right_skip
948                  {
949                      0.695\box_dp:N \strutbox
950                  }
951              \skip_set:Nn \l__enumext_minipage_after_skip
952                  {
953                      1.85\box_dp:N \strutbox
954                      + \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
955                  }
956          }
957      }
958      {
```

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.

```
959          \skip_if_eq:nnTF
960              { \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip } } { \c_zero_skip }
961              {
962                  \skip_set:Nn \l__enumext_minipage_left_skip
963                      {
964                          0.5\box_dp:N \strutbox
965                          - \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
966                      }
967                  \skip_set:Nn \l__enumext_minipage_right_skip
968                      {
969                          \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
970                      }
971                  \skip_set:Nn \l__enumext_minipage_after_skip
972                      {
973                          1.6\box_dp:N \strutbox
974                      }
975              }
976              {
977                  \skip_set:Nn \l__enumext_minipage_left_skip
978                      {
979                          0.5875\box_dp:N \strutbox
980                          - \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
981                      }
982                  \skip_set:Nn \l__enumext_minipage_right_skip
983                      {
984                          + \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
985                          + \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
986                      }
987                  \skip_set:Nn \l__enumext_minipage_after_skip
988                      {
989                          0.325\box_dp:N \strutbox
990                          + \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
991                      }
992              }
993          }
994      }
```

(*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.

```
995  \cs_new_protected:Nn \__enumext_zero_parsep:
996      {
```

```
997      \int_case:nn { \l__enumext_level_int }
998        {
999          { 2 }{
1000             \skip_if_eq:nnF { \l__enumext_parsep_i_skip } { \c_zero_skip }
1001               {
1002                 \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
1003               }
1004           }
1005          { 3 }{
1006             \skip_if_eq:nnF { \l__enumext_parsep_ii_skip } { \c_zero_skip }
1007               {
1008                 \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
1009               }
1010           }
1011          { 4 }{
1012             \skip_if_eq:nnF { \l__enumext_parsep_iii_skip } { \c_zero_skip }
1013               {
1014                 \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
1015               }
1016           }
1017       }
1018    }
```

(*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*.

```
1019 \cs_new_protected:Nn \__enumext_mini_addvspace:
1020    {
1021       \__enumext_mini_set_vskip:
1022       \mode_if_vertical:T
1023         {
1024           \skip_add:Nn \l__enumext_minipage_left_skip
1025             {
1026               \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
1027             }
1028           \skip_add:Nn \l__enumext_minipage_after_skip
1029             {
1030               \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
1031             }
1032         }
1033       \par\nopagebreak
1034       \addvspace { \l__enumext_minipage_left_skip }
1035    }
```

(*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.

```
1036 \cs_new_protected:Nn \__enumext_keyans_mini_set_vskip:
1037    {
1038       \skip_zero_new:N \l__enumext_minipage_after_skip
1039       \skip_zero_new:N \l__enumext_minipage_left_skip
1040       \skip_zero_new:N \l__enumext_minipage_right_skip
1041       \int_compare:nNnTF { \l__enumext_columns_v_int } > { 1 }
1042         {
1043           \skip_if_eq:nnTF { \l__enumext_topsep_v_skip } { \c_zero_skip }
1044             {
1045               \skip_set:Nn \l__enumext_minipage_left_skip { -0.25\box_dp:N \strutbox }
1046               \skip_set:Nn \l__enumext_minipage_right_skip { 0.705\box_dp:N \strutbox }
1047               \skip_set:Nn \l__enumext_minipage_after_skip { \box_dp:N \strutbox }
1048               \skip_if_eq:nnF { \l__enumext_parsep_i_skip } { \c_zero_skip }
1049                 {
1050                   \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
1051                 }
1052             }
```

```
1053              {
1054                \skip_set:Nn \l__enumext_minipage_left_skip
1055                  {
1056                    \skip_use:N \l__enumext_topsep_v_skip
1057                  }
1058                \skip_set:Nn \l__enumext_minipage_right_skip
1059                  {
1060                    0.705\box_dp:N \strutbox
1061                  }
1062                \skip_set:Nn \l__enumext_minipage_after_skip
1063                  {
1064                    1.85\box_dp:N \strutbox + \l__enumext_topsep_v_skip
1065                  }
1066              }
1067          }
1068          {
1069            \skip_if_eq:nnTF { \l__enumext_topsep_v_skip } { \c_zero_skip }
1070              {
1071                \skip_set:Nn \l__enumext_minipage_left_skip
1072                  {
1073                    0.5\box_dp:N \strutbox
1074                    + \l__enumext_partopsep_v_skip
1075                  }
1076                \skip_set:Nn \l__enumext_minipage_right_skip
1077                  {
1078                    \l__enumext_partopsep_v_skip
1079                  }
1080                \skip_set:Nn \l__enumext_minipage_after_skip { 1.6\box_dp:N \strutbox }
1081              }
1082              {
1083                \skip_set:Nn \l__enumext_minipage_left_skip
1084                  {
1085                    0.5875\box_dp:N \strutbox - \l__enumext_partopsep_v_skip
1086                  }
1087                \skip_set:Nn \l__enumext_minipage_right_skip
1088                  {
1089                    \l__enumext_topsep_v_skip + \l__enumext_partopsep_v_skip
1090                  }
1091                \skip_set:Nn \l__enumext_minipage_after_skip
1092                  {
1093                    0.325\box_dp:N \strutbox + \l__enumext_topsep_v_skip
1094                  }
1095              }
1096          }
1097      }
```

(*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.

```
1098  \cs_new_protected:Nn \__enumext_keyans_mini_addvspace:
1099    {
1100      \__enumext_keyans_mini_set_vskip:
1101      \mode_if_vertical:T
1102        {
1103          \skip_add:Nn \l__enumext_minipage_left_skip
1104            {
1105              \l__enumext_partopsep_v_skip
1106            }
1107          \skip_add:Nn \l__enumext_minipage_after_skip
1108            {
1109              \l__enumext_partopsep_v_skip
1110            }
1111        }
1112      \par\nopagebreak
1113      \addvspace { \l__enumext_minipage_left_skip }
1114    }
```

(*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:`
`\__enumext_mini_set_vskip_viii:`

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

```
1115  \cs_new_protected:Nn \__enumext_mini_set_vskip_vii:
1116  {
1117    \skip_zero_new:N \l__enumext_minipage_left_skip
1118    \skip_gzero_new:N \g__enumext_minipage_right_skip
1119    \skip_gzero_new:N \g__enumext_minipage_after_skip
1120    \skip_if_eq:nnTF { \l__enumext_topsep_vii_skip } { \c_zero_skip }
1121      {
1122        \skip_set:Nn \l__enumext_minipage_left_skip { 0.5\box_dp:N \strutbox }
1123        \skip_gset:Nn \g__enumext_minipage_right_skip { 0.325\box_dp:N \strutbox }
1124      }
1125      {
1126        \skip_set:Nn \l__enumext_minipage_left_skip { 0.5875\box_dp:N \strutbox }
1127        \skip_gset:Nn \g__enumext_minipage_right_skip
1128          {
1129            \l__enumext_topsep_vii_skip
1130          }
1131        \skip_gset:Nn \g__enumext_minipage_after_skip
1132          {
1133            0.325\box_dp:N \strutbox + \l__enumext_topsep_vii_skip
1134          }
1135      }
1136  }
1137  \cs_new_protected:Nn \__enumext_mini_set_vskip_viii:
1138  {
1139    \skip_zero_new:N \l__enumext_minipage_after_skip
1140    \skip_zero_new:N \l__enumext_minipage_left_skip
1141    \skip_zero_new:N \l__enumext_minipage_right_skip
1142    \skip_if_eq:nnTF { \l__enumext_topsep_viii_skip } { \c_zero_skip }
1143      {
1144        \skip_set:Nn \l__enumext_minipage_left_skip
1145          {
1146            0.5\box_dp:N \strutbox
1147          }
1148        \skip_set:Nn \l__enumext_minipage_right_skip
1149          {
1150            \l__enumext_partopsep_viii_skip
1151          }
1152        \skip_set:Nn \l__enumext_minipage_after_skip
1153          {
1154            1.6\box_dp:N \strutbox
1155          }
1156      }
1157      {
1158        \skip_set:Nn \l__enumext_minipage_left_skip
1159          {
1160            0.5875\box_dp:N \strutbox
1161          }
1162        \skip_set:Nn \l__enumext_minipage_right_skip
1163          {
1164            \l__enumext_topsep_viii_skip
1165          }
1166        \skip_set:Nn \l__enumext_minipage_after_skip
1167          {
1168            0.325\box_dp:N \strutbox + \l__enumext_topsep_viii_skip
1169          }
1170      }
1171  }
```

(*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.

```
1172  \cs_new_protected:Nn \__enumext_mini_addvspace_vii:
```

```
1173   {
1174       \__enumext_mini_set_vskip_vii:
1175       \par\nopagebreak
1176       \addvspace { \l__enumext_minipage_left_skip }
1177   }
1178 \cs_new_protected:Nn \__enumext_mini_addvspace_viii:
1179   {
1180       \__enumext_mini_set_vskip_viii:
1181       \par\nopagebreak
1182       \addvspace { \l__enumext_minipage_left_skip }
1183   }
```

(*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.

```
1184 \NewDocumentCommand \miniright { s }
1185   {
1186       \int_compare:nNnT { \l__enumext_keyans_pic_level_int } = { 1 }
1187         {
1188             \msg_error:nnn { enumext } { wrong-miniright-place }
1189         }
1190       \int_compare:nNnT { \l__enumext_level_int } = { 0 }
1191         {
1192             \msg_error:nnn { enumext } { wrong-miniright-place }
1193         }
1194       \int_compare:nNnTF { \l__enumext_keyans_level_int } = { 1 }
1195         {
1196             \__enumext_keyans_mini_right_cmd:n {#1}
1197         }
1198         { \__enumext_mini_right_cmd:n {#1} }
1199   }
```

(*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.

```
1200 \cs_new_protected:Npn \__enumext_mini_right_cmd:n #1
1201   {
1202       \dim_compare:nNnTF
1203         { \dim_use:c { l__enumext_minipage_right_ \__enumext_level: _dim } } > { \c_zero_dim }
1204         {
1205             \__enumext_multicols_stop:
1206             \end{__enumext_mini_env*}
1207             \hfill
1208             \begin{__enumext_mini_env*}
1209               { \dim_use:c { l__enumext_minipage_right_ \__enumext_level: _dim } }
1210               \par\addvspace { \l__enumext_minipage_right_skip }
1211               \bool_if:nF {#1}
1212                 {
1213                     \centering
1214                 }
1215               \int_gzero:N \g__enumext_minipage_stat_int
1216         }
1217         { \msg_error:nnn { enumext } { wrong-miniright-use } }
1218   }
```

(*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.

```
1219 \cs_new_protected:Npn \__enumext_keyans_mini_right_cmd:n #1
1220   {
1221     \dim_compare:nNnTF { \l__enumext_minipage_right_v_dim } > { \c_zero_dim }
1222       {
1223         \__enumext_keyans_multicols_stop:
1224         \end{__enumext_mini_env*}
1225         \hfill
1226         \begin{__enumext_mini_env*}{ \l__enumext_minipage_right_v_dim }
1227           \par\addvspace { \l__enumext_minipage_right_skip }
1228           \bool_if:nF {#1}
1229             {
1230               \centering
1231             }
1232           \int_gzero:N \g__enumext_minipage_stat_int
1233       }
1234       { \msg_error:nnn { enumext } { wrong-miniright-use } }
1235   }
```

(*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.

```
1236 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1237   {
1238     \keys_define:nn { enumext / #1 }
1239       {
1240         above  .skip_set:c = { l__enumext_vspace_above_#2_skip },
1241         above  .value_required:n = true,
1242         above* .code:n     = \bool_set_true:c { l__enumext_vspace_a_star_#2_bool }
1243                               \keys_set:nn { enumext / #1 } { above = {##1} },
1244         above* .value_required:n = true,
1245         below  .skip_set:c = { l__enumext_vspace_below_#2_skip },
1246         below  .value_required:n = true,
1247         below* .code:n     = \bool_set_true:c { l__enumext_vspace_b_star_#2_bool }
1248                               \keys_set:nn { enumext / #1 } { below = {##1} },
1249         below* .value_required:n = true,
1250       }
1251   }
1252 \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.

```
1253 \cs_new_protected:Nn \__enumext_vspace_above:
1254   {
1255     \skip_if_eq:nnF
1256       { \skip_use:c { l__enumext_vspace_above_ \__enumext_level: _skip } } { \c_zero_skip }
1257       {
1258         \bool_if:cTF { l__enumext_vspace_a_star_ \__enumext_level: _bool }
1259           {
1260             \vspace*{ \skip_use:c { l__enumext_vspace_above_ \__enumext_level: _skip } }
1261           }
1262           {
1263             \vspace { \skip_use:c { l__enumext_vspace_above_ \__enumext_level: _skip } }
1264           }
1265       }
1266   }
```

(*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.

```
1267 \cs_new_protected:Nn \__enumext_vspace_below:
1268   {
1269     \skip_if_eq:nnF
1270       { \skip_use:c { l__enumext_vspace_below_ \__enumext_level: _skip } } { \c_zero_skip }
1271       {
1272         \bool_if:cTF { l__enumext_vspace_b_star_ \__enumext_level: _bool }
1273           {
1274             \vspace*{ \skip_use:c { l__enumext_vspace_below_ \__enumext_level: _skip } }
1275           }
1276           {
1277             \vspace { \skip_use:c { l__enumext_vspace_below_ \__enumext_level: _skip } }
1278           }
1279       }
1280   }
```

(*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.

```
1281 \cs_new_protected:Nn \__enumext_vspace_above_v:
1282   {
1283     \skip_if_eq:nnF { \l__enumext_vspace_above_v_skip } { \c_zero_skip }
1284       {
1285         \bool_if:NTF \l__enumext_vspace_a_star_v_bool
1286           {
1287             \vspace*{ \l__enumext_vspace_above_v_skip }
1288           }
1289           { \vspace { \l__enumext_vspace_above_v_skip } }
1290       }
1291   }
```

(*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.

```
1292 \cs_new_protected:Nn \__enumext_vspace_below_v:
1293   {
1294     \skip_if_eq:nnF { \l__enumext_vspace_below_v_skip } { \c_zero_skip }
1295       {
1296         \bool_if:NTF \l__enumext_vspace_b_star_v_bool
1297           {
1298             \vspace*{ \l__enumext_vspace_below_v_skip }
1299           }
1300           { \vspace { \l__enumext_vspace_below_v_skip } }
1301       }
1302   }
```

(*End of definition for* __enumext_vspace_below_v:.)

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

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

```
1303 \cs_new_protected:Nn \__enumext_vspace_above_vii:
1304   {
1305     \skip_if_eq:nnF { \l__enumext_vspace_above_vii_skip } { \c_zero_skip }
1306       {
1307         \bool_if:NTF \l__enumext_vspace_a_star_vii_bool
1308           {
1309             \vspace*{ \l__enumext_vspace_above_vii_skip }
1310           }
1311           { \vspace { \l__enumext_vspace_above_vii_skip } }
1312       }
1313   }
1314 \cs_new_protected:Nn \__enumext_vspace_above_viii:
1315   {
1316     \skip_if_eq:nnF { \l__enumext_vspace_above_viii_skip } { \c_zero_skip }
```

```
1317        {
1318            \bool_if:NTF \l__enumext_vspace_a_star_viii_bool
1319                {
1320                    \vspace*{ \l__enumext_vspace_above_viii_skip }
1321                }
1322                { \vspace { \l__enumext_vspace_above_viii_skip } }
1323        }
1324    }
```

(*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.

```
1325  \cs_new_protected:Nn \__enumext_vspace_below_vii:
1326    {
1327        \skip_if_eq:nnF { \l__enumext_vspace_below_vii_skip } { \c_zero_skip }
1328            {
1329                \bool_if:NTF \l__enumext_vspace_b_star_vii_bool
1330                    {
1331                        \vspace*{ \l__enumext_vspace_below_vii_skip }
1332                    }
1333                    { \vspace { \l__enumext_vspace_below_vii_skip } }
1334            }
1335    }
1336  \cs_new_protected:Nn \__enumext_vspace_below_viii:
1337    {
1338        \skip_if_eq:nnF { \l__enumext_vspace_below_viii_skip } { \c_zero_skip }
1339            {
1340                \bool_if:NTF \l__enumext_vspace_b_star_viii_bool
1341                    {
1342                        \vspace*{ \l__enumext_vspace_below_viii_skip }
1343                    }
1344                    { \vspace { \l__enumext_vspace_below_viii_skip } }
1345            }
1346    }
```

(*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*.

```
1347  \cs_set_protected:Npn \__enumext_tmp:n #1
1348    {
1349        \keys_define:nn { enumext / #1 }
1350            {
1351                series  .str_set:N = \l__enumext_series_str,
1352                series  .value_required:n = true,
1353                resume  .code:n = \__enumext_resume_series:n {##1},
1354                resume* .code:n = \__enumext_resume_starred:,
1355                resume* .value_forbidden:n = true,
1356            }
1357    }
1358  \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.

```
1359  \cs_new:Npn \__enumext_filter_series:n #1
1360    {
1361        \use:e
1362            {
1363                \keyval_parse:NNn
1364                    \__enumext_filter_series_key:n
1365                    \__enumext_filter_series_pair:nn {#1}
1366            }
1367    }
```

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.

```
1368 \cs_new:Npn \__enumext_filter_series_key:n #1
1369   {
1370     \str_case:nnF {#1}
1371       {
1372         { resume } {}
1373         { resume* } {}
1374       }
1375       { , { \exp_not:n {#1} } }
1376   }
```

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.

```
1377 \cs_new:Npn \__enumext_filter_series_pair:nn #1#2
1378   {
1379     \str_case:nnF {#1}
1380       {
1381         { series } {}
1382         { resume } {}
1383         { start } {}
1384         { save-ans } {}
1385         { save-key } {}
1386       }
1387       { , { \exp_not:n {#1} } = { \exp_not:n {#2} } }
1388   }
```

(*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 (§10.33) and to the function `\__enumext_parse_keys_vii:n` in the enumext* environment definition (§10.36).

```
1389 \cs_new_protected:Npn \__enumext_parse_series:n #1
1390   {
1391     \str_if_empty:NTF \l__enumext_series_str
1392       {
1393         \bool_if:NF \l__enumext_resume_active_bool
1394           {
1395             \__enumext_resume_last:n {#1}
1396           }
1397       }
1398       {
1399         \tl_gclear_new:c { g__enumext_series_ \l__enumext_series_str _tl }
1400         \tl_gset:ce { g__enumext_series_ \l__enumext_series_str _tl }
1401           { \__enumext_filter_series:n {#1} }
1402         \int_if_exist:cF { g__enumext_series_ \l__enumext_series_str _int }
1403           {
1404             \int_new:c { g__enumext_series_ \l__enumext_series_str _int }
1405           }
1406       }
1407   }
```

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.

```
1408 \cs_new_protected:Npn \__enumext_resume_last:n #1
1409   {
1410     \bool_if:NT \l__enumext_standar_level_one_bool
1411       {
1412         \tl_gclear:N \g__enumext_standar_series_tl
1413         \tl_gset:Ne \g__enumext_standar_series_tl { \__enumext_filter_series:n {#1} }
```

```
1414              }
1415          \bool_if:NT \l__enumext_starred_level_one_bool
1416              {
1417                  \tl_gclear:N \g__enumext_starred_series_tl
1418                  \tl_gset:Ne \g__enumext_starred_series_tl { \__enumext_filter_series:n {#1} }
1419              }
1420      }
```

(*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 (§10.33)
and the enumext* environment definition (§10.36).

```
1421  \cs_new_protected:Nn \__enumext_resume_save_counter:
1422    {
1423      \bool_if:NT \g__enumext_standar_bool
1424        {
1425          \tl_if_empty:NF \l__enumext_series_str
1426            {
1427              \int_gset_eq:cN
1428                { g__enumext_series_ \l__enumext_series_str _int } \value{enumXi}
1429            }
1430          \tl_if_empty:NTF \l__enumext_resume_name_tl
1431            {
1432              \str_if_empty:NT \l__enumext_series_str
1433                {
1434                  \int_gset_eq:NN \g__enumext_resume_int \value{enumXi}
1435                }
1436            }
1437            {
1438              \int_if_exist:cT { g__enumext_series_ \l__enumext_resume_name_tl _int }
1439                {
1440                  \int_gset_eq:cN
1441                    { g__enumext_series_ \l__enumext_resume_name_tl _int } \value{enumXi}
1442                }
1443            }
1444          \int_if_exist:cT { g__enumext_resume_ \l__enumext_store_name_tl _int }
1445            {
1446              \int_gset_eq:cN
1447                { g__enumext_resume_ \l__enumext_store_name_tl _int } \value{enumXi}
1448            }
1449        }
1450      \bool_if:NT \g__enumext_starred_bool
1451        {
1452          \tl_if_empty:NF \l__enumext_series_str
1453            {
1454              \int_gset_eq:cN
1455                { g__enumext_series_ \l__enumext_series_str _int } \value{enumXvii}
1456            }
1457          \tl_if_empty:NTF \l__enumext_resume_name_tl
1458            {
1459              \str_if_empty:NT \l__enumext_series_str
1460                {
1461                  \int_gset_eq:NN \g__enumext_resume_vii_int \value{enumXvii}
1462                }
1463            }
1464            {
1465              \int_if_exist:cT { g__enumext_series_ \l__enumext_resume_name_tl _int }
1466                {
1467                  \int_gset_eq:cN
1468                    { g__enumext_series_ \l__enumext_resume_name_tl _int } \value{enumXvii}
1469                }
```

```
1470              }
1471          \int_if_exist:cT { g__enumext_resume_ \l__enumext_store_name_tl _int }
1472              {
1473                \int_gset_eq:cN
1474                  { g__enumext_resume_ \l__enumext_store_name_tl _int } \value{enumXvii}
1475              }
1476          }
1477      }
```

(*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 \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.

```
1478  \cs_new_protected:Npn \__enumext_resume_series:n #1
1479    {
1480      \tl_if_empty:nTF {#1}
1481        {
1482          \__enumext_resume_counter:n { }
1483        }
1484        {
1485          \tl_if_exist:cTF { g__enumext_series_ \tl_to_str:n {#1} _tl }
1486            {
1487              \__enumext_resume_counter:n {#1}
1488              \bool_if:NT \g__enumext_standar_bool
1489                {
1490                  \keys_set:nv { enumext / level-1 }
1491                    { g__enumext_series_ \tl_to_str:n {#1} _tl }
1492                }
1493              \bool_if:NT \g__enumext_starred_bool
1494                {
1495                  \keys_set:nv { enumext / enumext* }
1496                    { g__enumext_series_ \tl_to_str:n {#1} _tl }
1497                }
1498            }
1499            {
1500              \bool_if:NT \g__enumext_standar_bool
1501                {
1502                  \msg_error:nnn { enumext } { unknown-series } {#1}
1503                }
1504              \bool_if:NT \g__enumext_starred_bool
1505                {
1506                  \msg_error:nnn { enumext } { unknown-series } {#1}
1507                }
1508            }
1509        }
1510    }
```

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

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

```
1511  \cs_new_protected:Npn \__enumext_resume_counter:n #1
1512    {
1513      \bool_set_true:N \l__enumext_resume_active_bool
1514      \tl_set:Nn \l__enumext_resume_name_tl {#1}
1515      \tl_if_empty:NTF \l__enumext_resume_name_tl
1516        {
1517          \__enumext_resume_counter:
1518        }
```

```
1519        {
1520            \__enumext_resume_counter_series:
1521        }
1522      \__enumext_resume_counter_save_ans:
1523    }
```

The __enumext_resume_counter: function is executed when the resume key is used *without value*, only the counters for the *"first level"* of the environments will be set.

```
1524  \cs_new_protected:Nn \__enumext_resume_counter:
1525    {
1526      \bool_if:NT \g__enumext_standar_bool
1527        {
1528            \int_gincr:N \g__enumext_resume_int
1529            \int_set_eq:NN \l__enumext_start_i_int \g__enumext_resume_int
1530        }
1531      \bool_if:NT \g__enumext_starred_bool
1532        {
1533            \int_gincr:N \g__enumext_resume_vii_int
1534            \int_set_eq:NN \l__enumext_start_vii_int \g__enumext_resume_vii_int
1535        }
1536    }
```

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.

```
1537  \cs_new_protected:Nn \__enumext_resume_counter_series:
1538    {
1539      \bool_if:NT \g__enumext_standar_bool
1540        {
1541            \int_set:Nn \l__enumext_start_i_int
1542                {
1543                    \int_use:c { g__enumext_series_ \l__enumext_resume_name_tl _int } + 1
1544                }
1545        }
1546      \bool_if:NT \g__enumext_starred_bool
1547        {
1548            \int_set:Nn \l__enumext_start_vii_int
1549                {
1550                    \int_use:c { g__enumext_series_ \l__enumext_resume_name_tl _int } + 1
1551                }
1552        }
1553    }
```

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.

```
1554  \cs_new_protected:Nn \__enumext_resume_counter_save_ans:
1555    {
1556      \bool_lazy_and:nnT
1557        { \bool_if_p:N \l__enumext_standar_level_one_bool }
1558        { \bool_if_p:N \l__enumext_store_active_bool }
1559        {
1560            \int_set:Nn \l__enumext_start_i_int
1561                {
1562                    \int_use:c { g__enumext_resume_ \l__enumext_store_name_tl _int } + 1
1563                }
1564        }
1565      \bool_lazy_and:nnT
1566        { \bool_if_p:N \l__enumext_starred_level_one_bool }
1567        { \bool_if_p:N \l__enumext_store_active_bool }
1568        {
1569            \int_set:Nn \l__enumext_start_vii_int
1570                {
1571                    \int_use:c { g__enumext_resume_ \l__enumext_store_name_tl _int } + 1
1572                }
1573        }
1574    }
```

(*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.

```
1575 \cs_new_protected:Nn \__enumext_resume_starred:
1576   {
1577     \bool_if:NT \g__enumext_standar_bool
1578       {
1579         \tl_if_empty:NF \g__enumext_standar_series_tl
1580           {
1581             \__enumext_resume_counter:n { }
1582             \keys_set:nV { enumext / level-1 } \g__enumext_standar_series_tl
1583           }
1584       }
1585     \bool_if:NT \g__enumext_starred_bool
1586       {
1587         \tl_if_empty:NF \g__enumext_starred_series_tl
1588           {
1589             \__enumext_resume_counter:n { }
1590             \keys_set:nV { enumext / enumext* } \g__enumext_starred_series_tl
1591           }
1592       }
1593   }
```

(*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*.

```
1594 \cs_set_protected:Npn \__enumext_tmp:n #1
1595   {
1596     \keys_define:nn { enumext / #1 }
1597       {
1598         save-ans .code:n = \__enumext_storing_set:n {##1},
1599         save-ans .value_required:n = true,
1600       }
1601   }
1602 \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
__enumext_storing_exec:

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

```
1603 \cs_new_protected:Npn \__enumext_storing_set:n #1
1604   {
1605     \tl_set:Ne \l__enumext_store_name_tl {#1}
1606     \tl_if_empty:NTF \l__enumext_store_name_tl
1607       {
1608         \bool_if:NT \l__enumext_standar_level_one_bool
1609           {
1610             \msg_error:nnnV
1611               { enumext } { save-ans-empty } { enumext } \g__enumext_standar_star_env_int
1612           }
1613         \bool_if:NT \l__enumext_starred_level_one_bool
1614           {
1615             \msg_error:nnnV
1616               { enumext } { save-ans-empty } { enumext* } \g__enumext_starred_star_env_int
1617           }
1618       }
1619       {
1620         \bool_if:NT \l__enumext_standar_level_one_bool
1621           {
1622             \msg_note:nnnee
```

```
1623              { enumext } { save-ans-ok } { enumext }
1624              { \l__enumext_store_name_tl } { \int_use:N \g__enumext_standar_star_env_int }
1625          \__enumext_storing_exec:
1626        }
1627      \bool_if:NT \l__enumext_starred_level_one_bool
1628        {
1629          \msg_note:nnnee
1630            { enumext } { save-ans-ok } { enumext* }
1631            { \l__enumext_store_name_tl } { \int_use:N \g__enumext_starred_star_env_int }
1632          \__enumext_storing_exec:
1633        }
1634    }
1635  }
```

The function `\__enumext_storing_exec:` will set to true the variable `\l__enumext_store_active_-bool` which activates the use of the `\anskey` command and the `keyans`, `keyans*` and `keyanspic` environments and will set to true the variable `\l__enumext_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*`.

```
1636  \cs_new_protected:Nn \__enumext_storing_exec:
1637    {
1638      \bool_set_true:N \l__enumext_store_active_bool
1639      \bool_set_true:N \l__enumext_store_ans_bool
1640      \prop_if_exist:cF { g__enumext_ \l__enumext_store_name_tl _prop }
1641        {
1642          \prop_new:c { g__enumext_ \l__enumext_store_name_tl _prop }
1643        }
1644      \seq_if_exist:cF { g__enumext_ \l__enumext_store_name_tl _seq }
1645        {
1646          \seq_new:c { g__enumext_ \l__enumext_store_name_tl _seq }
1647        }
1648      \int_if_exist:cF { g__enumext_resume_ \l__enumext_store_name_tl _int }
1649        {
1650          \int_new:c { g__enumext_resume_ \l__enumext_store_name_tl _int }
1651        }
1652    }
```

(*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.

```
1653 \cs_set_protected:Npn \__enumext_tmp:n #1
1654   {
1655     \keys_define:nn { enumext / #1 }
1656       {
1657         check-ans .bool_set:N = \l__enumext_check_ans_bool,
1658         check-ans .initial:n  = false,
1659         no-store  .code:n = {
1660                             \bool_set_false:N \l__enumext_store_ans_bool
1661                             \bool_set_false:N \l__enumext_check_ans_bool
1662                             },
1663         no-store  .value_forbidden:n = true,
1664       }
1665   }
1666 \clist_map_inline:nn
1667   {
1668     level-1, level-2, level-3, level-4, enumext*
1669   }
1670   { \__enumext_tmp:n {#1} }
```

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

10.23.2 Set-up check answer mechanism

__enumext_check_ans_set:

The function __enumext_check_ans_set: will adjust the value of the variable \g__enumext_count_-item_number_int by decrementing its value by one each time you open a nested level enumext environment.

```
1671 \cs_new_protected:Nn \__enumext_check_ans_set:
1672   {
1673     \int_case:nn { \l__enumext_level_int }
1674       {
1675         { 1 }{
1676                 \bool_lazy_all:nT
1677                   {
1678                     { \bool_if_p:N \g__enumext_starred_bool }
1679                     { \int_compare_p:nNn { \l__enumext_level_h_int } = { 1 } }
1680                   }
1681                   {
1682                     \int_gdecr:N \g__enumext_count_item_number_int
1683                     \typeout{ENUMEXT ~ STANDAR ~ NEEEEEEEEEEEESTED}
1684                   }
1685               }
1686         { 2 }{
1687                 \int_gdecr:N \g__enumext_count_item_number_int
1688               }
1689         { 3 }{
1690                 \int_gdecr:N \g__enumext_count_item_number_int
1691               }
1692         { 4 }{
1693                 \int_gdecr:N \g__enumext_count_item_number_int
1694               }
1695       }
1696     \int_case:nn { \l__enumext_level_h_int }
1697       {
1698         { 1 }{
1699                 \bool_if:NT \g__enumext_standar_bool
1700                   {
1701                     \int_gdecr:N \g__enumext_count_item_number_int
1702                     \typeout{ENUMEXT ~ STARRED ~ NEEEEEEEEEEEESTED}
1703                   }
1704               }
1705       }
1706   }
```

(*End of definition for* __enumext_check_ans_set:.)

`\__enumext_check_ans_exec:`

The function `\__enumext_check_ans_exec:` will count the number of times the `\item` and `\item*` commands appears per level within the `enumext` environment. The boolean variable `\l__enumext_store_ans_bool` controlled by the `no-store` key will increment the integer variable of the level counter by `1` to preserve the equality that we will use in the final comparison of the process.

```
1707 \cs_new_protected:Nn \__enumext_check_ans_exec:
1708   {
1709     \bool_if:NT \l__enumext_check_ans_bool
1710       {
1711         \__enumext_check_ans_set:
1712       }
1713   }
```

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

`\__enumext_check_ans_show:`

The function `\__enumext_check_ans_show:` compares all executions of `\item` and `\item*` with the executions of `\anskey`. After the function is executed, we set the integer variables to zero.

```
1714 \cs_new_protected:Nn \__enumext_check_ans_show:
1715   {
1716     \int_compare:nNnTF
1717       { \g__enumext_count_item_number_int } = { \g__enumext_count_item_anskey_int }
1718       {
1719         \msg_term:nnV { enumext } { items-same-answer } \g__enumext_store_name_tl
1720       }
1721       {
1722         \msg_warning:nnV { enumext } { item-different-answer } \g__enumext_store_name_tl
1723       }
1724     \int_gzero:N \g__enumext_count_item_number_int
1725     \int_gzero:N \g__enumext_count_item_anskey_int
1726   }
```

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

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*`.

```
1727 \cs_set_protected:Npn \__enumext_tmp:n #1
1728   {
1729     \keys_define:nn { enumext / #1 }
1730       {
1731         wrap-ans    .cs_set_protected:Np = \__enumext_anskey_wrapper:n ##1,
1732         wrap-ans    .initial:n = \fbox{##1},
1733         wrap-ans    .value_required:n = true,
1734         wrap-opt    .cs_set_protected:Np = \__enumext_keyans_wrapper_opt:n ##1,
1735         wrap-opt    .initial:n = [{##1}],
1736         wrap-opt    .value_required:n = true,
1737         save-sep    .tl_set:N  = \l__enumext_store_keyans_item_opt_sep_tl,
1738         save-sep    .initial:n = {, ~ },
1739         save-sep    .value_required:n = true,
1740         mark-ans    .tl_set:N  = \l__enumext_mark_answer_sym_tl,
1741         mark-ans    .initial:n = \textasteriskcentered,
1742         mark-ans    .value_required:n = true,
1743         mark-pos    .choice:,
1744         mark-pos  / left    .code:n = \str_set:Nn \l__enumext_mark_position_str { l },
1745         mark-pos  / right   .code:n = \str_set:Nn \l__enumext_mark_position_str { r },
1746         mark-pos    .initial:n = right,
1747         mark-pos    .value_required:n = true,
1748         show-ans    .bool_set:N = \l__enumext_show_answer_bool,
1749         show-ans    .initial:n  = false,
1750         show-ans    .value_required:n = true,
1751         show-pos    .bool_set:N = \l__enumext_show_position_bool,
1752         show-pos    .initial:n  = false,
1753         show-pos    .value_required:n = true,
1754         mark-ref    .tl_set:N   = \l__enumext_mark_ref_sym_tl,
1755         mark-ref    .initial:n  = \textasteriskcentered,
1756         mark-ref    .value_required:n = true,
1757         save-ref    .bool_set:N = \l__enumext_store_ref_key_bool,
1758         save-ref    .initial:n  = false,
1759         save-ref    .value_required:n = true,
1760       }
```

```
1761      }
1762  \clist_map_inline:nn { level-1, enumext* } { \__enumext_tmp:n {#1} }
```

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

mark-pos
show-ans

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

```
1763  \cs_set_protected:Npn \__enumext_tmp:n #1
1764    {
1765      \keys_define:nn { enumext / #1 }
1766        {
1767          mark-pos .choice:,
1768          mark-pos / left  .code:n = \str_set:Nn \l__enumext_mark_position_str { l },
1769          mark-pos / right .code:n = \str_set:Nn \l__enumext_mark_position_str { r },
1770          mark-pos .initial:n = right,
1771          mark-pos .value_required:n  = true,
1772          show-ans .bool_set:N = \l__enumext_show_answer_bool,
1773          show-ans .initial:n  = false,
1774          show-ans .value_required:n = true,
1775          show-pos .bool_set:N = \l__enumext_show_position_bool,
1776          show-pos .initial:n  = false,
1777          show-pos .value_required:n = true,
1778        }
1779    }
1780  \clist_map_inline:nn { keyans, keyans* } { \__enumext_tmp:n {#1} }
```

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

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.

```
1781  \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1782    {
1783      \keys_define:nn { enumext / #1 }
1784        {
1785          columns*      .code:n = \bool_set_true:c { l__enumext_store_columns_#2_bool }
1786                                  \int_set:cn { l__enumext_store_columns_#2_int } {##1}
1787                                  \tl_put_right:ce { l__enumext_store_opt_#2_tl }
1788                                    {
1789                                      columns = \exp_not:v { l__enumext_store_columns_#2_int },
1790                                    },
1791          columns*      .value_required:n  = true,
1792          columns-sep* .code:n = \bool_set_true:c { l__enumext_store_columns_sep_#2_bool }
1793                                  \dim_set:cn { l__enumext_store_columns_sep_#2_dim } {##1}
1794                                  \tl_put_right:ce { l__enumext_store_opt_#2_tl }
1795                                    {
1796                                      columns-sep = \exp_not:v { l__enumext_store_columns_sep_#2_di
1797                                    },
1798          columns-sep* .value_required:n  = true,
1799        }
1800    }
1801  \clist_map_inline:nn
1802    {
1803      {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {enumext*}{vii}
1804    }
1805    { \__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.

```
1806  \cs_generate_variant:Nn \prop_gput_if_not_in:Nnn { cen }
1807  \cs_new_protected:Npn \__enumext_store_addto_prop:n #1
1808    {
1809      \prop_gput_if_not_in:cen { g__enumext_ \l__enumext_store_name_tl _prop }
```

```
1810      {
1811        \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } + 1 }
1812      }
1813      { #1 }
1814    }
1815  \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.

```
1816  \cs_new_protected:Npn \__enumext_store_addto_seq:n #1
1817    {
1818      \seq_gput_right:cn { g__enumext_ \l__enumext_store_name_tl _seq } { #1 }
1819    }
1820  \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.

```
1821  \cs_new_protected:Nn \__enumext_store_level_open:
1822    {
1823      \bool_if:NT \l__enumext_store_ans_bool
1824        {
1825          \tl_if_empty:cTF { l__enumext_store_opt_ \__enumext_level: _tl }
1826            {
1827              \__enumext_store_addto_seq:n
1828                {
1829                  \item \begin{enumext}
1830                }
1831            }
1832            {
1833              \tl_put_left:cn { l__enumext_store_opt_ \__enumext_level: _tl }
1834                {
1835                  \item \begin{enumext} [
1836                }
1837              \tl_put_right:cn { l__enumext_store_opt_ \__enumext_level: _tl }
1838                {
1839                  ]
1840                }
1841              \__enumext_store_addto_seq:v { l__enumext_store_opt_ \__enumext_level: _tl }
1842            }
1843        }
1844    }
1845  \cs_new_protected:Nn \__enumext_store_level_close:
1846    {
1847      \bool_if:NT \l__enumext_store_ans_bool
1848        {
1849          \__enumext_store_addto_seq:n { \end{enumext} }
1850        }
1851    }
```

(*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.

```
1852  \cs_new_protected:Nn \__enumext_store_level_open_vii:
1853    {
1854      \bool_if:NT \l__enumext_store_ans_bool
1855        {
1856          \tl_if_empty:NTF \l__enumext_store_opt_vii_tl
1857            {
1858              \__enumext_store_addto_seq:n
1859                {
1860                  \item \mode_leave_vertical:
1861                    \vspace { -\skip_eval:n { \baselineskip + \parsep } }
1862                    \begin{enumext*}[before={\setlength{\topsep}{0pt}},]
1863                }
1864            }
1865            {
1866              \tl_put_left:Nn \l__enumext_store_opt_vii_tl
1867                {
1868                  \item \mode_leave_vertical:
1869                    \vspace { -\skip_eval:n { \baselineskip + \parsep } }
1870                    \begin{enumext*}[before={\setlength{\topsep}{0pt}},
1871                }
1872              \tl_put_right:Nn \l__enumext_store_opt_vii_tl
1873                {
1874                  ]
1875                }
1876              \__enumext_store_addto_seq:V \l__enumext_store_opt_vii_tl
1877            }
1878        }
1879    }
1880  \cs_new_protected:Nn \__enumext_store_level_close_vii:
1881    {
1882      \bool_if:NT \l__enumext_store_ans_bool
1883        {
1884          \__enumext_store_addto_seq:n { \end{enumext*} }
1885        }
1886    }
```

(*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`

```
1887  \cs_new_protected:Nn \__enumext_print_keyans_box:NN
1888    {
1889      \mode_leave_vertical:
1890      \skip_horizontal:n { -\dim_use:N #2 }
1891      \makebox[0pt][ r ]
1892        {
1893          \makebox[ \dim_use:N #1 ][ \l__enumext_mark_position_str ]
1894            {
1895              \tl_use:N \l__enumext_mark_answer_sym_tl
1896            }
1897        }
1898      \skip_horizontal:n { \dim_use:N #2 }
1899    }
1900  \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.

```
1901  \keys_define:nn { enumext / anskey }
1902    {
1903      item-sym* .tl_set:N   = \l__enumext_store_item_symbol_tl,
1904      item-sym* .value_required:n = true,
1905      item-pos* .dim_set:N  = \l__enumext_store_item_symbol_sep_dim,
1906      item-pos* .value_required:n = true,
1907      store-brk .bool_set:N = \l__enumext_store_columns_break_bool,
1908      store-brk .default:n  = true,
1909      store-brk .value_forbidden:n = true,
1910    }
```

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*⟩.

```
1911  \NewDocumentCommand \anskey { d() s o +m }
1912    {
1913      \bool_if:NF \l__enumext_store_active_bool
1914        {
1915          \msg_error:nnnn { enumext } { anskey-wrong-place }{ anskey }{ enumext }
1916        }
1917      \int_compare:nNnT { \l__enumext_keyans_level_int } = { 1 }
1918        {
1919          \msg_error:nnnn { enumext } { command-wrong-place }{ anskey }{ keyans }
1920        }
1921      \int_compare:nNnT { \l__enumext_keyans_pic_level_int } = { 1 }
1922        {
1923          \msg_error:nnnn { enumext } { command-wrong-place }{ anskey }{ keyanspic }
1924        }
1925      \group_begin:
1926        \bool_if:NT \l__enumext_store_ans_bool
1927          {
1928            \bool_if:NT \l__enumext_check_ans_bool
1929              {
1930                \int_gincr:N \g__enumext_count_item_anskey_int
1931              }
1932            \__enumext_store_anskey_code:nnnn {#1} {#2} {#3} {#4}
1933          }
1934      \group_end:
1935    }
```

(*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.

```
1936  \cs_new_protected:Npn \__enumext_store_anskey_code:nnnn #1 #2 #3 #4
1937    {
1938      \__enumext_store_addto_prop:n {#4}
1939      \bool_if:NT \l__enumext_store_ref_key_bool
1940        {
1941          \__enumext_store_internal_ref:
1942        }
1943      \__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.

```
1944      \tl_clear:N \l__enumext_store_anskey_arg_tl
1945      \tl_if_novalue:nF {#3}
1946        {
1947          \keys_set:nn { enumext / anskey } {#3}
1948        }
1949      \bool_lazy_and:nnT
1950        { \bool_if_p:N \l__enumext_store_columns_break_bool }
1951        { \bool_not_p:n { \l__enumext_starred_bool } }
1952        {
```

```
1953            \tl_put_left:Nn \l__enumext_store_anskey_arg_tl { \columnbreak }
1954          }
1955        \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).

```
1956        \tl_if_novalue:nF {#1}
1957          {
1958            \int_set:Nn \l__enumext_store_columns_join_int {#1}
1959            \bool_if:NT \l__enumext_starred_bool
1960              {
1961                \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
1962                  {
1963                    ( \exp_not:V \l__enumext_store_columns_join_int )
1964                  }
1965              }
1966          }
```

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.

```
1967        \bool_if:nTF {#2}
1968          {
1969            \tl_put_right:Nn \l__enumext_store_anskey_arg_tl { * }
1970            \tl_if_empty:NF \l__enumext_store_item_symbol_tl
1971              {
1972                \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
1973                  {
1974                    [ \exp_not:V \l__enumext_store_item_symbol_tl ]
1975                  }
1976              }
1977            \dim_compare:nT
1978              {
1979                \l__enumext_store_item_symbol_sep_dim != \c_zero_dim
1980              }
1981              {
1982                \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
1983                  {
1984                    [ \exp_not:V \l__enumext_store_item_symbol_sep_dim ]
1985                  }
1986              }
1987            \tl_put_right:Nn \l__enumext_store_anskey_arg_tl {#4}
1988          }
1989          {
1990            \tl_put_right:Nn \l__enumext_store_anskey_arg_tl {#4}
1991          }
```

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*⟩.

```
1992        \bool_lazy_and:nnT
1993          { \bool_if_p:N \l__enumext_store_ref_key_bool }
1994          { \bool_if_p:N \l__enumext_hyperref_bool }
1995          {
1996            \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
1997              {
1998                \hfill \exp_not:N \hyperlink { \exp_not:V \l__enumext_newlabel_arg_one_tl }
1999                    { \exp_not:V \l__enumext_mark_ref_sym_tl }
2000              }
2001          }
2002        \__enumext_store_addto_seq:V \l__enumext_store_anskey_arg_tl
2003      }
```

(*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.

```
2004  \cs_new_protected:Nn \__enumext_store_internal_ref:
2005    {
2006      \cs_set_protected:Npn \__enumext_tmp:n ##1
2007        {
```

```
2008        \tl_set_eq:cc { l__enumext_label_copy_##1_tl } { l__enumext_label_##1_tl }
2009        \tl_reverse:c { l__enumext_label_copy_##1_tl }
2010        \tl_remove_once:cn { l__enumext_label_copy_##1_tl } { . }
2011        \tl_reverse:c { l__enumext_label_copy_##1_tl }
2012      }
2013    \clist_map_inline:nn { i, ii, iii, iv, vii } { \__enumext_tmp:n {##1} }
2014    \cs_set:Npn \__enumext_tmp:n ##1
2015      { . \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.

```
2016      \bool_lazy_all:nT
2017        {
2018          { \bool_if_p:N \g__enumext_starred_bool }
2019          { \int_compare_p:nNn { \l__enumext_level_int } = { \c_zero_int } }
2020        }
2021        {
2022          \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2023            { \tl_use:N \l__enumext_label_copy_vii_tl }
2024        }
2025      \bool_lazy_all:nT
2026        {
2027          { \bool_if_p:N \l__enumext_standar_bool }
2028          { \bool_if_p:N \g__enumext_starred_bool }
2029          { \int_compare_p:nNn { \l__enumext_level_int } > { \c_zero_int } }
2030        }
2031        {
2032          \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2033            {
2034              \tl_use:N \l__enumext_label_copy_vii_tl
2035              \int_step_function:nnN { 1 } { \l__enumext_level_int } \__enumext_tmp:n
2036            }
2037        }
```

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.

```
2038      \bool_lazy_all:nT
2039        {
2040          { \bool_if_p:N \l__enumext_standar_bool }
2041          { \int_compare_p:nNn { \l__enumext_level_int } > { \c_zero_int } }
2042          { \int_compare_p:nNn { \l__enumext_level_h_int } = { \c_zero_int } }
2043          { \bool_not_p:n { \l__enumext_starred_bool } }
2044        }
2045        {
2046          \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2047            {
2048              \tl_use:N \l__enumext_label_copy_i_tl
2049              \int_step_function:nnN { 2 } { \l__enumext_level_int } \__enumext_tmp:n
2050            }
2051        }
2052      \cs_set:Npn \__enumext_tmp:n ##1
2053        { \tl_use:c { l__enumext_label_copy_ \int_to_roman:n {##1} _tl } }
2054      \bool_lazy_all:nT
2055        {
2056          { \bool_if_p:N \l__enumext_standar_bool }
2057          { \int_compare_p:nNn { \l__enumext_level_int } > { \c_zero_int } }
2058          { \bool_not_p:n { \g__enumext_starred_bool } }
2059          { \int_compare_p:nNn { \l__enumext_level_h_int } > { \c_zero_int } }
2060        }
2061        {
2062          \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2063            {
2064              \int_step_function:nnN { 1 } { \l__enumext_level_int } \__enumext_tmp:n
2065              . \tl_use:N \l__enumext_label_copy_vii_tl
2066            }
2067        }
```

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

```
2068      \tl_put_right:Ne \l__enumext_newlabel_arg_one_tl
2069        {
2070          \l__enumext_store_name_tl \c_colon_str
```

```
2071        \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } }
2072      }
```

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.

```
2073      \tl_put_right:Ne \l__enumext_store_write_aux_file_tl
2074        {
2075          \__enumext_newlabel:nn
2076            { \exp_not:V \l__enumext_newlabel_arg_one_tl }
2077            { \l__enumext_newlabel_arg_two_tl }
2078        }
2079      \l__enumext_store_write_aux_file_tl
2080    }
```

(*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.

```
2081  \cs_new_protected:Npn \__enumext_store_anskey_show_wrap:n #1
2082    {
2083      \par
2084      \bool_if:NT \l__enumext_starred_bool
2085        {
2086          \cs_set:Nn \__enumext_level: { vii }
2087        }
2088      \__enumext_print_keyans_box:cc
2089        { l__enumext_labelwidth_ \__enumext_level: _dim }
2090        { l__enumext_labelsep_ \__enumext_level: _dim }
2091      \__enumext_anskey_wrapper:n { #1 }
2092    }
```

(*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.

```
2093  \cs_new_protected:Npn \__enumext_store_anskey_show_left:n #1
2094    {
2095      \bool_if:NT \l__enumext_show_answer_bool
2096        {
2097          \__enumext_store_anskey_show_wrap:n { #1 }
2098        }
2099      \bool_if:NT \l__enumext_show_position_bool
2100        {
2101          \tl_set:Ne \l__enumext_mark_answer_sym_tl
2102            {
2103              \group_begin:
2104              \exp_not:N \normalfont
2105              \exp_not:N \footnotesize [ \int_eval:n
2106                {
2107                  \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
2108                }
2109              ]
2110              \group_end:
2111            }
2112          \__enumext_store_anskey_show_wrap:n { #1 }
2113        }
2114    }
```

(*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.

```
2115  \cs_new_protected:Npn \__enumext_keyans_addto_prop:n #1
2116    {
2117      \tl_clear:N \l__enumext_store_keyans_label_tl
2118      \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
2119        {
2120          \tl_put_right:Ne \l__enumext_store_keyans_label_tl { \l__enumext_label_vi_tl }
2121        }
2122        {
2123          \tl_put_right:Ne \l__enumext_store_keyans_label_tl { \l__enumext_label_v_tl }
2124        }
2125      \tl_if_novalue:nF { #1 }
2126        {
2127          % Set save-sep
2128          \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_tl
2129            {
2130              \tl_put_right:Ne \l__enumext_store_keyans_label_tl { \l__enumext_store_keyans_item_op
2131            }
2132          \tl_put_right:Ne \l__enumext_store_keyans_label_tl { #1 }
2133        }
2134      \__enumext_store_addto_prop:V \l__enumext_store_keyans_label_tl
2135    }
```

(*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:
__enumext_keyans_store_ref_aux_i:
__enumext_keyans_store_ref_aux_ii:

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

```
2136  \cs_new_protected:Nn \__enumext_keyans_store_ref:
2137    {
2138      \bool_if:NT \l__enumext_store_ref_key_bool
2139        {
2140          \cs_set_protected:Npn \__enumext_tmp:n ##1
2141            {
2142              \tl_set_eq:cc { l__enumext_label_copy_##1_tl } { l__enumext_label_##1_tl }
2143              \tl_reverse:c { l__enumext_label_copy_##1_tl }
2144              \tl_remove_once:cn { l__enumext_label_copy_##1_tl } { . }
2145              \tl_reverse:c { l__enumext_label_copy_##1_tl }
2146            }
2147          \clist_map_inline:nn { i, v, vi, vii, viii } { \__enumext_tmp:n {##1} }
2148          \__enumext_keyans_store_ref_aux_i:
2149        }
2150    }
```

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.

```
2151  \cs_new_protected:Nn \__enumext_keyans_store_ref_aux_i:
2152    {
2153      \bool_if:NT \g__enumext_starred_bool
2154        {
2155          \tl_set_eq:NN \l__enumext_label_copy_i_tl \l__enumext_label_copy_vii_tl
2156        }
2157      \int_compare:nNnT { \l__enumext_keyans_pic_level_int } = { 1 }
2158        {
2159          \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2160            { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_vi_tl }
2161        }
2162      \int_compare:nNnT { \l__enumext_keyans_level_int } = { 1 }
2163        {
2164          \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2165            { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_v_tl }
2166        }
2167      \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
2168        {
2169          \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
```

```
2170              { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_viii_tl }
2171            }
2172        \tl_put_right:Ne \l__enumext_newlabel_arg_one_tl
2173          {
2174            \l__enumext_store_name_tl \c_colon_str
2175            \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } }
2176          }
2177        \__enumext_keyans_store_ref_aux_ii:
2178      }
```

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.

```
2179  \cs_new_protected:Nn \__enumext_keyans_store_ref_aux_ii:
2180    {
2181      \tl_put_right:Ne \l__enumext_store_write_aux_file_tl
2182        {
2183          \__enumext_newlabel:nn
2184            { \exp_not:V \l__enumext_newlabel_arg_one_tl }
2185            { \l__enumext_newlabel_arg_two_tl }
2186        }
2187      \l__enumext_store_write_aux_file_tl
2188    }
```

(*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

__enumext_keyans_addto_seq:n
__enumext_keyans_addto_seq_link:

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

```
2189  \cs_new_protected:Npn \__enumext_keyans_addto_seq:n #1
2190    {
2191      \tl_clear:N \l__enumext_store_keyans_label_tl
2192      \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
2193        {
2194          \tl_put_right:Ne \l__enumext_store_keyans_label_tl { \item \l__enumext_label_vi_tl }
2195        }
2196        {
2197          \tl_put_right:Ne \l__enumext_store_keyans_label_tl { \item \l__enumext_label_v_tl }
2198        }
2199      \tl_if_novalue:nF { #1 }
2200        {
2201          \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_tl
2202            {
2203              \tl_put_right:Ne \l__enumext_store_keyans_label_tl { \l__enumext_store_keyans_item_op
2204            }
2205          \tl_put_right:Ne \l__enumext_store_keyans_label_tl { #1 }
2206        }
2207      \__enumext_keyans_addto_seq_link:
2208    }
```

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_keyans_check_ans:nn and increment the value of the integer variable \g__enumext_count_item_anskey_int handled by the check-ans key.

```
2209  \cs_new_protected:Nn \__enumext_keyans_addto_seq_link:
2210    {
2211      \bool_lazy_and:nnT
2212        { \bool_if_p:N \l__enumext_store_ref_key_bool }
2213        { \bool_if_p:N \l__enumext_hyperref_bool }
2214        {
2215          \tl_put_right:Ne \l__enumext_store_keyans_label_tl
2216            {
2217              \hfill \exp_not:N \hyperlink
2218                {
2219                  \exp_not:V \l__enumext_newlabel_arg_one_tl
2220                }
```

```
2221                    { \exp_not:V \l__enumext_mark_ref_sym_tl }
2222                }
2223            }
2224        \__enumext_store_addto_seq:V \l__enumext_store_keyans_label_tl
2225        \tl_gset:NV \g__enumext_check_ans_item_tl \l__enumext_store_keyans_label_tl
2226        \bool_if:NT \l__enumext_check_ans_bool
2227            {
2228                \int_gincr:N \g__enumext_count_item_anskey_int
2229            }
2230    }
```

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

10.26.4   Check for starred commands

__enumext_keyans_check_ans:nn The function __enumext_keyans_check_ans:nn performs an extra check for the 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.

```
2231  \cs_new_protected:Npn \__enumext_keyans_check_ans:nn #1 #2
2232    {
2233        \tl_if_empty:NTF \g__enumext_check_ans_item_tl
2234            {
2235                \msg_warning:nnnn { enumext } { missing-starred }{ #1 }{ #2 }
2236            }
2237            { \tl_gclear:N \g__enumext_check_ans_item_tl }
2238    }
```

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

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

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

__enumext_keyans_show_left:n
__enumext_keyans_show_ans:
__enumext_keyans_show_pos:
__enumext_keyans_show_item_opt:

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

```
2239  \cs_new_protected:Npn \__enumext_keyans_show_left:n #1
2240    {
2241        \tl_if_novalue:nF { #1 }
2242            {
2243                \tl_set:Ne \l__enumext_keyans_item_opt_tl { #1 }
2244            }
2245        \bool_if:NT \l__enumext_show_answer_bool
2246            {
2247                \__enumext_keyans_show_ans:
2248            }
2249        \bool_if:NT \l__enumext_show_position_bool
2250            {
2251                \__enumext_keyans_show_pos:
2252            }
2253    }
2254  \cs_new_protected:Nn \__enumext_keyans_show_item_opt:
2255    {
2256        \tl_if_empty:NF \l__enumext_keyans_item_opt_tl
2257            {
2258                \bool_lazy_or:nnT
2259                  { \bool_if_p:N \l__enumext_show_answer_bool }
2260                  { \bool_if_p:N \l__enumext_show_position_bool }
2261                    {
2262                        \__enumext_keyans_wrapper_opt:n { \l__enumext_keyans_item_opt_tl } \c_space_tl
2263                    }
2264            }
2265    }
2266  \cs_new_protected:Nn \__enumext_keyans_show_ans:
2267    {
2268        \tl_put_left:Nn \l__enumext_label_v_tl
2269            {
2270                \__enumext_print_keyans_box:NN \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
2271            }
2272    }
2273  \cs_new_protected:Nn \__enumext_keyans_show_pos:
2274    {
```

```
2275        \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
2276          {
2277            \tl_set:Ne \l__enumext_mark_answer_sym_tl
2278              {
2279                \group_begin:
2280                \exp_not:N \normalfont
2281                \exp_not:N \footnotesize [ \int_eval:n
2282                  {
2283                    \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
2284                  }
2285                  ]
2286                \group_end:
2287              }
2288          }
2289          {
2290            \tl_set:Ne \l__enumext_mark_answer_sym_tl
2291              {
2292                \group_begin:
2293                \exp_not:N \normalfont
2294                \exp_not:N \footnotesize [ \int_eval:n
2295                  {
2296                    \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } + 1
2297                  }
2298                  ]
2299                \group_end:
2300              }
2301          }
2302        \tl_put_left:Nn \l__enumext_label_v_tl
2303          {
2304            \__enumext_print_keyans_box:NN
2305              \l__enumext_labelwidth_i_dim
2306              \l__enumext_labelsep_i_dim
2307          }
2308      }
```

(*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*`.

```
2309 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
2310   {
2311     \keys_define:nn { enumext / #1 }
2312       {
2313         item-sym* .tl_set:c  = { l__enumext_item_symbol_#2_tl },
2314         item-sym* .value_required:n = true,
2315         item-sym* .initial:n  = {$\star$},
2316         item-pos* .dim_set:c  = { l__enumext_item_symbol_sep_#2_dim },
2317         item-pos* .value_required:n = true,
2318       }
2319   }
2320 \clist_map_inline:nn
2321   {
2322     {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {enumext*}{vii}
2323   }
2324   { \__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.

```
2325 \cs_new_protected:Nn \__enumext_footnotetext:nn
2326   {
2327     \footnotetext[#1]{#2}
2328   }
2329 \cs_new_protected:Nn \__enumext_renew_footnote:
```

```
2330      {
2331        \seq_gclear:N \g__enumext_footnote_arg_seq
2332        \seq_gclear:N \g__enumext_footnote_int_seq
2333        \RenewDocumentCommand \footnote { o +m }
2334          {
2335            \tl_if_novalue:nTF {##1}
2336              {
2337                \stepcounter{footnote}
2338                \int_gset_eq:Nc \g__enumext_footnote_int { c@footnote }
2339              }
2340              {
2341                \int_gset:Nn \g__enumext_footnote_int { ##1 }
2342              }
2343            \footnotemark [ \g__enumext_footnote_int ]
2344            \seq_gput_right:Nn \g__enumext_footnote_arg_seq { ##2 }
2345            \seq_gput_right:NV \g__enumext_footnote_int_seq \g__enumext_footnote_int
2346          }
2347      }
2348 \cs_new_protected:Nn \__enumext_print_footnote:
2349      {
2350        \seq_if_empty:NF \g__enumext_footnote_int_seq
2351          {
2352            \seq_map_pairwise_function:NNN
2353              \g__enumext_footnote_int_seq
2354              \g__enumext_footnote_arg_seq
2355              \__enumext_footnotetext:nn
2356          }
2357      }
```

(*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).

```
2358 \cs_new_protected:Npn \__enumext_default_item:n #1
2359      {
2360        \tl_if_novalue:nTF {#1}
2361          {
2362            \bool_if:NT \l__enumext_check_ans_bool
2363              {
2364                \int_gincr:N \g__enumext_count_item_number_int
2365              }
2366            \bool_set_true:c { l__enumext_wrap_label_ \__enumext_level: _bool }
2367            \__enumext_item_std:w \tl_use:c { l__enumext_fake_item_indent_ \__enumext_level: _tl }
2368          }
2369          {
2370            \bool_set_eq:cc
2371              { l__enumext_wrap_label_ \__enumext_level: _bool }
2372              { l__enumext_wrap_label_opt_ \__enumext_level: _bool }
2373            \__enumext_item_std:w [#1] \tl_use:c { l__enumext_fake_item_indent_ \__enumext_level: _tl
2374          }
2375      }
```

(*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).

```
2376  \cs_new_protected:Npn \__enumext_starred_item:nn #1 #2
2377    {
2378      \tl_if_novalue:nF {#1}
2379        {
2380          \tl_set:cn { l__enumext_item_symbol_ \__enumext_level: _tl } {#1}
2381        }
2382      \tl_gset_eq:Nc \g__enumext_item_symbol_tl { l__enumext_item_symbol_ \__enumext_level: _tl }
2383      \tl_if_novalue:nTF {#2}
2384        {
2385          \dim_set_eq:cc
2386            { l__enumext_item_symbol_sep_ \__enumext_level: _dim }
2387            { l__enumext_labelsep_ \__enumext_level: _dim }
2388        }
2389        {
2390          \dim_set:cn { l__enumext_item_symbol_sep_ \__enumext_level: _dim } {#2}
2391        }
2392      \bool_if:NT \l__enumext_check_ans_bool
2393        {
2394          \int_gincr:N \g__enumext_count_item_number_int
2395        }
2396      \bool_set_true:c { l__enumext_wrap_label_ \__enumext_level: _bool }
2397      \__enumext_item_std:w \tl_use:c { l__enumext_fake_item_indent_ \__enumext_level: _tl }
2398    }
```

(*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 (§10.32).

```
2399  \cs_new_protected:Nn \__enumext_redefine_item:
2400    {
2401      \RenewDocumentCommand \item { s o o }
2402        {
2403          \bool_if:nTF {##1}
2404            {
2405              \__enumext_starred_item:nn {##2} {##3}
2406            }
2407            { \__enumext_default_item:n {##2} }
2408        }
2409    }
```

(*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`.

```
2410  \cs_new_protected:Npn \__enumext_keyans_default_item:n #1
2411    {
2412      \tl_if_novalue:nTF { #1 }
2413        {
```

```
2414          \bool_set_true:N \l__enumext_wrap_label_v_bool
2415          \__enumext_item_std:w \tl_use:N \l__enumext_fake_item_indent_v_tl
2416        }
2417        {
2418          \bool_set_eq:NN \l__enumext_wrap_label_v_bool \l__enumext_wrap_label_opt_v_bool
2419          \__enumext_item_std:w [#1] \tl_use:N \l__enumext_fake_item_indent_v_tl
2420        }
2421      }
```

(*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*⟩.

```
2422 \cs_new_protected:Npn \__enumext_keyans_starred_item:n #1
2423   {
2424     \tl_set_eq:NN \l__enumext_keyans_tmpa_tl \l__enumext_label_v_tl
2425     \__enumext_keyans_show_left:n { #1 }
2426     \bool_set_true:N \l__enumext_wrap_label_v_bool
2427     \__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*⟩.

```
2428     \tl_set_eq:NN \l__enumext_label_v_tl \l__enumext_keyans_tmpa_tl
2429     \__enumext_keyans_addto_prop:n { #1 }
2430     \__enumext_keyans_store_ref:
2431     \__enumext_keyans_addto_seq:n { #1 }
2432   }
```

(*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 (§10.32).

```
2433 \cs_new_protected:Nn \__enumext_keyans_redefine_item:
2434   {
2435     \RenewDocumentCommand \item { s o }
2436       {
2437         \bool_if:nTF {##1}
2438           {
2439             \peek_remove_spaces:n
2440               {
2441                 \__enumext_keyans_starred_item:n {##2}
2442               }
2443           }
2444           {
2445             \__enumext_keyans_default_item:n {##2}
2446           }
2447       }
2448   }
```

(*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.

```
2449 \cs_new_protected:Nn \__enumext_item_starred:
2450   {
2451     \tl_if_empty:cF { l__enumext_item_symbol_ \__enumext_level: _tl }
2452       {
2453         \mode_leave_vertical:
```

```
2454        \skip_horizontal:n { -\dim_use:c { l__enumext_item_symbol_sep_ \__enumext_level: _dim } }
2455        \makebox[ 0pt ][ r ]{ \g__enumext_item_symbol_tl }
2456        \skip_horizontal:n { \dim_use:c { l__enumext_item_symbol_sep_ \__enumext_level: _dim } }
2457      }
2458    }
```

(*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 (§10.32).

```
2459 \cs_new_protected:Nn \__enumext_make_label:
2460   {
2461     \RenewDocumentCommand \makelabel { m }
2462       {
2463         \tl_use:c { l__enumext_label_fill_left_ \__enumext_level: _tl }
2464         \tl_use:c { l__enumext_label_font_style_ \__enumext_level: _tl }
2465         \bool_if:cTF { l__enumext_wrap_label_ \__enumext_level: _bool }
2466           {
2467             \__enumext_item_starred:
2468             \use:c { __enumext_wrapper_label_ \__enumext_level: :n } { ##1 }
2469           }
2470           { ##1 }
2471         \tl_use:c { l__enumext_label_fill_right_ \__enumext_level: _tl }
2472         \tl_gclear:N \g__enumext_item_symbol_tl
2473       }
2474   }
```

(*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 (§10.32).

```
2475 \cs_new_protected:Nn \__enumext_keyans_make_label:
2476   {
2477     \RenewDocumentCommand \makelabel { m }
2478       {
2479         \tl_use:N \l__enumext_label_fill_left_v_tl
2480         \tl_use:N \l__enumext_label_font_style_v_tl
2481         \bool_if:NTF \l__enumext_wrap_label_v_bool
2482           {
2483             \__enumext_wrapper_label_v:n { ##1 }
2484           }
2485           { ##1 }
2486         \tl_use:N \l__enumext_label_fill_right_v_tl
2487       }
2488   }
```

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

10.31 Calculation of \leftmargin and \itemindent

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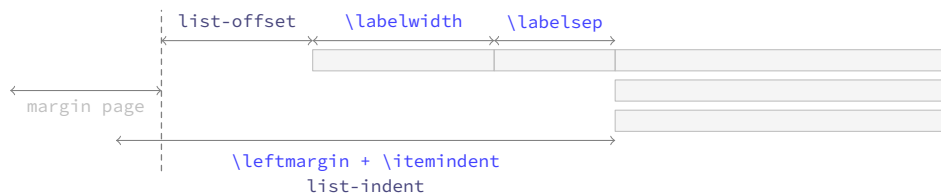
Figure 9: Representation of standard horizontal lengths in list environment.

The idea is to have control over these margins so that our list does not overlap the left margin of the page. The *key* relationship is that the right edge of the \labelsep equals the right edge of the \itemindent, so that the left edge of the *label box* is at \leftmargin+\itemindent minus \labelwidth+\labelsep. Thus, the handling of the margins by the package will be as shown in the figure 10.
Where the default values will look like in the figure 11.

__enumext_calc_hspace:NNNNNNN
__enumext_calc_hspace:ccccccc

The function __enumext_calc_hspace:NNNNNNN takes seven arguments to be able to determine horizontal spaces for all list environment:
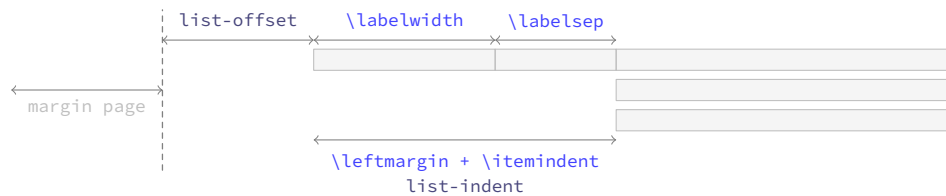
Figure 10: Representation of horizontal lengths concept in list in enumext.
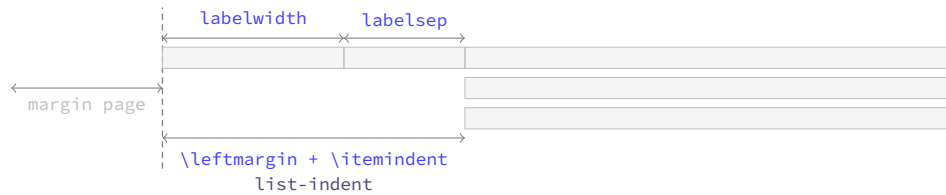
Figure 11: Default horizontal lengths in enumext.

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

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

This function is passed to __enumext_list_arg_two_X: which is used in the definition of the enumext and keyans environments (§10.32).

```
2489  \cs_new_protected:Npn \__enumext_calc_hspace:NNNNNNN #1 #2 #3 #4 #5 #6 #7
2490    {
2491      \dim_compare:nNnT { #1 } < { \c_zero_dim }
2492        {
2493          \msg_warning:nnnV { enumext } { width-non-positive }{ labelwidth }{ #1 }
2494          \dim_set:Nn #1 { \dim_abs:n { #1 } }
2495        }
2496      \dim_compare:nNnT { #2 } < { \c_zero_dim }
2497        {
2498          \msg_warning:nnnV { enumext } { width-negative }{ labelsep }{ #2 }
2499          \dim_set:Nn #2 { \dim_abs:n { #2 } }
2500        }
```

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

```
2501      \bool_if:nF #7 { \dim_set:Nn #4 { #1 + #2} }
```

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

```
2502      \dim_compare:nNnTF { #4 } < { \c_zero_dim }
2503        {
2504          \dim_set:Nn #6 { #1 + #2 - #4}
2505          \dim_set:Nn #5 { #1 + #2 + #3 - #6 }
2506        }
2507        {
2508          \dim_compare:nNnT { #4 } = { #1 + #2 }
2509            { \dim_set:Nn #6 { \c_zero_dim } }
2510          \dim_compare:nNnT { #4 } < { #1 + #2 }
2511            { \dim_set:Nn #6 { #1 + #2 - #4} }
2512          \dim_compare:nNnT { #4 } > { #1 + #2 }
2513            {
2514              \dim_set:Nn #6 { -#1 - #2 + #4}
2515              \dim_set:Nn #6 { #6*-1}
2516            }
2517          \dim_set:Nn #5 { #1 + #2 + #3 - #6 }
2518        }
2519    }
2520  \cs_generate_variant:Nn \__enumext_calc_hspace:NNNNNNN { ccccccc }
```

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

10.32 Setting second argument of the lists

At this point of the code we have already programmed 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`.

`\__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:`

In this function for the second list argument we will implement the keys `start`, `resume` and `show-length` together with the redefinition of `\item` for `enumext` and `keyans` environments.
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.

```
2521 \cs_set_protected:Npn \__enumext_tmp:n #1
2522   {
2523     \cs_new_protected:cpn { __enumext_list_arg_two_#1: }
2524       {
2525         \__enumext_calc_hspace:ccccccc
2526           { l__enumext_labelwidth_#1_dim } { l__enumext_labelsep_#1_dim }
2527           { l__enumext_listoffset_#1_dim } { l__enumext_leftmargin_tmp_#1_dim }
2528           { l__enumext_leftmargin_#1_dim } { l__enumext_itemindent_#1_dim }
2529           { l__enumext_leftmargin_tmp_#1_bool }
2530         \clist_map_inline:nn
2531           { labelsep, labelwidth, itemindent, leftmargin, rightmargin, listparindent }
2532           { \dim_set_eq:cc {####1} { l__enumext_####1_#1_dim } }
2533         \clist_map_inline:nn { topsep, parsep, partopsep, itemsep }
2534           { \skip_set_eq:cc {####1} { l__enumext_####1_#1_skip } }
2535         \usecounter { enumX#1 }
2536         \setcounter { enumX#1 } { \int_eval:n { \int_use:c { l__enumext_start_#1_int } - 1 } }
2537         \str_if_eq:nnTF {#1} { v }
2538           {
2539             \__enumext_keyans_redefine_item:
2540             \__enumext_keyans_make_label:
2541             \__enumext_keyans_fake_item:
2542             \bool_if:cT { l__enumext_show_length_#1_bool }
2543               {
2544                 \msg_term:nnnn { enumext } { list-lengths-not-nested } { v } { keyans }
2545               }
2546           }
2547           {
2548             \__enumext_redefine_item:
2549             \__enumext_make_label:
2550             \__enumext_use_key_ref:
2551             \__enumext_fake_item:
2552             \bool_if:cT { l__enumext_show_length_#1_bool }
2553               {
2554                 \msg_term:nnne { enumext } { list-lengths } {#1} { \int_use:N \l__enumext_level_i
2555               }
2556           }
2557       }
2558   }
2559 \clist_map_inline:nn { i, ii, iii, iv, v } { \__enumext_tmp:n {#1} }
```

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

`\__enumext_list_arg_two_vii:`
`\__enumext_list_arg_two_viii:`

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

```
2560 \cs_set_protected:Npn \__enumext_tmp:n #1
2561   {
2562     \cs_new_protected:cpn { __enumext_list_arg_two_#1: }
2563       {
2564         \__enumext_calc_hspace:ccccccc
2565           { l__enumext_labelwidth_#1_dim } { l__enumext_labelsep_#1_dim }
2566           { l__enumext_listoffset_#1_dim } { l__enumext_leftmargin_tmp_#1_dim }
2567           { l__enumext_leftmargin_#1_dim } { l__enumext_itemindent_#1_dim }
2568           { l__enumext_leftmargin_tmp_#1_bool }
2569         \clist_map_inline:nn
2570           { labelsep, labelwidth, itemindent, leftmargin, rightmargin, listparindent }
2571           { \dim_set_eq:cc {####1} { l__enumext_####1_#1_dim } }
2572         \clist_map_inline:nn { topsep, parsep, partopsep, itemsep }
2573           { \skip_set_eq:cc {####1} { l__enumext_####1_#1_skip } }
```

```
2574        \skip_set_eq:Nc \parsep  { l__enumext_itemsep_#1_skip }
2575        \skip_zero:N \partopsep
2576        \usecounter { enumX#1 }
2577        \setcounter { enumX#1 } { \int_eval:n { \int_use:c { l__enumext_start_#1_int } - 1 } }
2578        \__enumext_use_key_ref_h:
2579        \str_if_eq:nnTF {#1} { vii }
2580          {
2581            \__enumext_fake_item_vii:
2582            \bool_if:cT { l__enumext_show_length_vii_bool }
2583              { \msg_term:nnnn { enumext } { list-lengths-not-nested } { vii } { enumext* } }
2584          }
2585          {
2586            \__enumext_fake_item_viii:
2587            \bool_if:cT { l__enumext_show_length_#1_bool }
2588              { \msg_term:nnnn { enumext } { list-lengths-not-nested } { #1 } { keyans* } }
2589          }
2590      }
2591    }
2592  \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:.)

10.33   The environment enumext

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

```
2593  \NewDocumentEnvironment{enumext}{ O{} }
2594    {
2595      \__enumext_safe_exec:
2596      \__enumext_parse_keys:n {#1}
2597      \__enumext_before_list:
2598      \__enumext_start_store_level:
2599      \__enumext_start_list:nn
2600        { \tl_use:c { l__enumext_label_ \__enumext_level: _tl } }
2601        {
2602          \use:c { __enumext_list_arg_two_ \__enumext_level: : }
2603          \__enumext_before_keys_exec:
2604        }
2605      \__enumext_after_args_exec:
2606    }
2607    {
2608      \__enumext_stop_list:
2609      \__enumext_stop_store_level:
2610      \__enumext_after_list:
2611    }
```

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

__enumext_safe_exec: First we will execute the function __enumext_current_env_set_bool: 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_level_one_bool to *"true"* only if the environment is not nested and we are at the first level of it.

```
2612  \cs_new_protected:Nn \__enumext_safe_exec:
2613    {
2614      \__enumext_current_env_set_bool:
2615      \int_incr:N \l__enumext_level_int
2616      \int_compare:nNnT { \l__enumext_level_int } > { 4 }
2617        { \msg_fatal:nn { enumext } { list-too-deep } }
2618      \bool_set_true:N \l__enumext_standar_bool
2619      \bool_lazy_all:nT
2620        {
2621          { \bool_if_p:N \g__enumext_standar_bool }
2622          { \int_compare_p:nNn { \l__enumext_level_int } = { 1 } }
2623          { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
2624        }
2625        {
2626          %%\typeout{[[ON-FIRST-LEVEL-ENUMEXT-NOT-NESTED]]}
2627          \bool_set_true:N \l__enumext_standar_level_one_bool
2628        }
2629    }
```

(*End of definition for* __enumext_safe_exec:.)

`\__enumext_parse_keys:n`

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

```
2630  \cs_new_protected:Npn \__enumext_parse_keys:n #1
2631    {
2632      \tl_if_novalue:nF {#1}
2633        {
2634          \str_clear:N \l__enumext_series_str
2635          \int_compare:nNnTF { \l__enumext_level_int } = { 1 }
2636            {
2637              \keys_set:nn { enumext / level-1 } {#1}
2638              \__enumext_parse_series:n {#1}
2639            }
2640            {
2641              \exp_args:Ne \keys_set:nn
2642                { enumext / level-\int_use:N \l__enumext_level_int } {#1}
2643            }
2644          \bool_if:NT \l__enumext_store_active_bool
2645            {
2646              \__enumext_parse_store_keys:n {#1}
2647            }
2648        }
2649    }
```

(*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:`.

```
2650  \cs_new_protected:Npn \__enumext_parse_store_keys:n #1
2651    {
2652      \bool_if:cF { l__enumext_store_columns_ \__enumext_level: _bool }
2653        {
2654          \regex_match:nnT { \b columns\b } {#1}
2655            {
2656              \int_set_eq:cc
2657                { l__enumext_store_columns_ \__enumext_level: _int }
2658                { l__enumext_columns_  \__enumext_level: _int }
2659              \tl_put_right:ce { l__enumext_store_opt_ \__enumext_level: _tl }
2660                {
2661                  columns = \exp_not:v { l__enumext_store_columns_ \__enumext_level: _int },
2662                }
2663            }
2664        }
2665      \bool_if:cF { l__enumext_store_columns_sep_ \__enumext_level: _bool }
2666        {
2667          \regex_match:nnT { \b columns-sep \b} {#1}
2668            {
2669              \dim_set_eq:cc
2670                { l__enumext_store_columns_sep_ \__enumext_level: _dim }
2671                { l__enumext_columns_sep_  \__enumext_level: _dim }
2672              \tl_put_right:ce { l__enumext_store_opt_ \__enumext_level: _tl }
2673                {
2674                  columns-sep = \exp_not:v { l__enumext_store_columns_sep_ \__enumext_level: _dim }
2675                }
2676            }
2677        }
2678    }
```

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

`\__enumext_start_store_level:`
`\__enumext_stop_store_level:`

The `\__enumext_start_store_level:` and `\__enumext_stop_store_level:` functions activate the 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.

```
2679  \cs_new_protected:Nn \__enumext_start_store_level:
2680    {
2681      \bool_lazy_all:nT
2682        {
```

```
2683          { \bool_if_p:N \l__enumext_store_active_bool }
2684          { \bool_not_p:n { \l__enumext_keyans_env_bool } }
2685          { \bool_not_p:n { \g__enumext_starred_bool } }
2686        }
2687        {
2688          \int_compare:nNnT { \l__enumext_level_int } > { 1 }
2689            {
2690              \bool_set_true:c { l__enumext_store_upper_level_ \__enumext_level: _bool }
2691              \__enumext_store_level_open:
2692            }
2693        }
2694      \bool_lazy_all:nT
2695        {
2696          { \bool_if_p:N \l__enumext_store_active_bool }
2697          { \bool_not_p:n { \l__enumext_keyans_env_bool } }
2698          { \bool_if_p:N \g__enumext_starred_bool }
2699        }
2700        {
2701          \int_compare:nNnT { \l__enumext_level_int } > { 0 }
2702            {
2703              \bool_set_true:c { l__enumext_store_upper_level_ \__enumext_level: _bool }
2704              \__enumext_store_level_open:
2705            }
2706        }
2707    }
2708  \cs_new_protected:Nn \__enumext_stop_store_level:
2709    {
2710      \bool_if:cT { l__enumext_store_upper_level_ \__enumext_level: _bool }
2711        {
2712          \__enumext_store_level_close:
2713        }
2714    }
```

(*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.

```
2715  \cs_new_protected:Nn \__enumext_before_list:
2716    {
2717      \__enumext_vspace_above:
2718      \__enumext_before_args_exec:
```

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

```
2719      \__enumext_check_ans_exec:
```

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.

```
2720      \dim_compare:nNnT
2721        { \dim_use:c { l__enumext_minipage_right_ \__enumext_level: _dim } } > { \c_zero_dim }
2722        {
2723          \dim_set:cn { l__enumext_minipage_left_ \__enumext_level: _dim }
2724            {
2725              \linewidth
2726              - \dim_use:c { l__enumext_minipage_right_ \__enumext_level: _dim }
2727              - \dim_use:c { l__enumext_minipage_hsep_ \__enumext_level: _dim }
2728            }
```

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*.

```
2729          \bool_set_true:c { l__enumext_minipage_active_ \__enumext_level: _bool }
2730          \int_gincr:N \g__enumext_minipage_stat_int
2731          \__enumext_mini_addvspace:
```

```
2732        \nointerlineskip\noindent
2733        \begin{__enumext_mini_env*}
2734          { \dim_use:c { l__enumext_minipage_left_ \__enumext_level: _dim } }
2735        }
2736      \__enumext_multicols_start:
2737    }
```

(*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.

```
2738  \cs_new_protected:Nn \__enumext_multicols_start:
2739    {
2740      \int_compare:nNnT
2741        { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } > { 1 }
2742        {
2743          \dim_compare:nNnT
2744            { \dim_use:c { l__enumext_columns_sep_ \__enumext_level: _dim } } = { \c_zero_dim }
2745            {
2746              \dim_set:cn { l__enumext_columns_sep_ \__enumext_level: _dim }
2747                {
2748                  ( \dim_use:c { l__enumext_labelwidth_ \__enumext_level: _dim }
2749                    + \dim_use:c { l__enumext_labelsep_ \__enumext_level: _dim }
2750                  ) / \int_use:c { l__enumext_columns_ \__enumext_level: _int }
2751                  - \dim_use:c { l__enumext_listoffset_ \__enumext_level: _dim }
2752                }
2753            }
2754          \dim_set_eq:Nc \columnsep { l__enumext_columns_sep_ \__enumext_level: _dim  }
2755          \skip_zero:N \multicolsep
2756          \int_compare:nNnT { \l__enumext_level_int } > { 1 }
2757            {
2758              \dim_zero:N \columnseprule
2759            }
```

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.

```
2760          \bool_if:cF { l__enumext_minipage_active_ \__enumext_level: _bool }
2761            {
2762              \__enumext_multi_addvspace:
2763            }
2764          \raggedcolumns
2765          \begin{multicols}{ \int_use:c { l__enumext_columns_ \__enumext_level: _int } }
2766        }
2767    }
```

(*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.

```
2768  \cs_new_protected:Nn \__enumext_multicols_stop:
2769    {
2770      \int_compare:nNnT
2771        { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } > { 1 }
2772        {
2773          \end{multicols}
2774          \bool_if:cF { l__enumext_minipage_active_ \__enumext_level: _bool }
2775            {
2776              \par\addvspace{ \skip_use:c { l__enumext_multicols_below_ \__enumext_level: _skip } }
2777            }
2778        }
2779    }
```

(*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.

```
2780 \cs_new_protected:Nn \__enumext_after_list:
2781   {
2782     \bool_if:cTF { l__enumext_minipage_active_ \__enumext_level: _bool }
2783       {
2784         \int_compare:nNnT { \g__enumext_minipage_stat_int } = { 1 }
2785           {
2786             \msg_warning:nn { enumext } { missing-miniright }
2787             \miniright
2788           }
2789         \int_gzero:N \g__enumext_minipage_stat_int
2790         \end{__enumext_mini_env*}
2791         \par\addvspace { \l__enumext_minipage_after_skip }
2792       }
2793       { \__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.

```
2794     \bool_lazy_and:nnT
2795       { \bool_if_p:N \l__enumext_check_ans_bool }
2796       { \bool_not_p:n { \g__enumext_starred_bool } }
2797       {
2798         \bool_gset_true:N \g__enumext_check_ans_show_bool
2799         \tl_gset:NV \g__enumext_store_name_tl \l__enumext_store_name_tl
2800       }
```

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.

```
2801     \__enumext_after_stop_list:
2802     \__enumext_vspace_below:
2803     \bool_set_false:N \l__enumext_standar_bool
2804     \__enumext_resume_save_counter:
2805   }
```

(*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.

```
2806 \__enumext_after_env:nn {enumext}
2807   {
2808     \int_compare:nNnT { \l__enumext_level_int } = { 0 }
2809       {
2810         \bool_if:NT \g__enumext_check_ans_show_bool
2811           {
2812             \__enumext_check_ans_show:
2813           }
2814         \bool_gset_false:N \g__enumext_standar_bool
2815         \bool_gset_false:N \g__enumext_check_ans_show_bool
2816         \tl_gclear:N \g__enumext_store_name_tl
2817       }
2818   }
```

10.34 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.

```
2819 \NewDocumentEnvironment{keyans}{ O{} }
2820   {
2821     \__enumext_keyans_safe_exec:
2822     \__enumext_keyans_parse_keys:n {#1}
2823     \__enumext_before_list_v:
2824     \__enumext_start_list:nn
2825       { \tl_use:N \l__enumext_label_v_tl }
2826       {
```

```
2827                 \__enumext_list_arg_two_v:
2828                 \__enumext_before_keys_exec_v:
2829               }
2830           \__enumext_after_args_exec_v:
2831         }
2832         {
2833           \__enumext_keyans_check_ans:nn { item }{ keyans }
2834           \__enumext_stop_list:
2835           \__enumext_after_list_v:
2836         }
```

(*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.

```
2837 \cs_new_protected:Nn \__enumext_keyans_safe_exec:
2838   {
2839     \bool_if:NF \l__enumext_store_active_bool
2840       {
2841         \msg_error:nnnn { enumext } { wrong-place }{ keyans }{ save-ans }
2842       }
2843     \int_incr:N \l__enumext_keyans_level_int
2844     \bool_set_true:N \l__enumext_keyans_env_bool
2845     % Set false for interfering with enumext nested in keyans (yes, its possible and crayze)
2846     \bool_set_false:N \l__enumext_store_active_bool
2847     \int_compare:nNnT { \l__enumext_keyans_level_int } > { 1 }
2848       {
2849         \msg_error:nn { enumext } { keyans-nested }
2850       }
2851     \int_compare:nNnT { \l__enumext_level_int } > { 1 }
2852       {
2853         \msg_error:nn { enumext } { keyans-wrong-level }
2854       }
2855   }
```

(*End of definition for* __enumext_keyans_safe_exec:.)

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

```
2856 \cs_new_protected:Npn \__enumext_keyans_parse_keys:n #1
2857   {
2858     \keys_set:nn { enumext / keyans } {#1}
2859   }
```

(*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.

```
2860 \cs_new_protected:Nn \__enumext_before_list_v:
2861   {
2862     \__enumext_vspace_above_v:
2863     \__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.

```
2864     \dim_compare:nNnT { \l__enumext_minipage_right_v_dim } > { \c_zero_dim }
2865       {
2866         \dim_set:Nn \l__enumext_minipage_left_v_dim
2867           {
2868             \linewidth - \l__enumext_minipage_right_v_dim - \l__enumext_minipage_hsep_v_dim
2869           }
```

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*.

```
2870          \bool_set_true:N \l__enumext_minipage_active_v_bool
2871          \int_gincr:N \g__enumext_minipage_stat_int
2872          \__enumext_keyans_mini_addvspace:
2873          \nointerlineskip\noindent
2874          \begin{__enumext_mini_env*}{ \l__enumext_minipage_left_v_dim }
2875       }
```

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

```
2876       \__enumext_keyans_multicols_start:
2877    }
```

(*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.

```
2878 \cs_new_protected:Nn \__enumext_keyans_multicols_start:
2879   {
2880      \int_compare:nNnT { \l__enumext_columns_v_int } > { 1 }
2881        {
```

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

```
2882          \dim_compare:nNnT { \l__enumext_columns_sep_v_dim } = { \c_zero_dim }
2883            {
2884              \dim_set:Nn \l__enumext_columns_sep_v_dim
2885                {
2886                  (
2887                    \l__enumext_labelwidth_v_dim + \l__enumext_labelsep_v_dim
2888                  ) / \l__enumext_columns_v_int
2889                  - \l__enumext_listoffset_v_dim
2890                }
2891            }
2892          \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).

```
2893          \skip_zero:N \multicolsep
2894          \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.

```
2895          \bool_if:NF \l__enumext_minipage_active_v_bool
2896            {
2897              \__enumext_keyans_multi_addvspace:
2898            }
2899          \raggedcolumns
2900          \begin{multicols}{ \l__enumext_columns_v_int }
2901        }
2902    }
```

(*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.

```
2903 \cs_new_protected:Nn \__enumext_keyans_multicols_stop:
2904   {
2905      \int_compare:nNnT { \l__enumext_columns_v_int } > { 1 }
2906        {
2907          \end{multicols}
2908          \bool_if:NF \l__enumext_minipage_active_v_bool
2909            {
2910              \par\addvspace{ \l__enumext_multicols_below_v_skip }
2911            }
2912        }
2913    }
```

(*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.

```
2914 \cs_new_protected:Nn \__enumext_after_list_v:
2915   {
2916     \bool_if:NTF \l__enumext_minipage_active_v_bool
2917       {
2918         \int_compare:nNnT { \g__enumext_minipage_stat_int } = { 1 }
2919           {
2920             \msg_warning:nn { enumext } { missing-miniright }
2921             \miniright
2922           }
2923         \int_gzero:N \g__enumext_minipage_stat_int
2924         \end{__enumext_mini_env*}
2925         \par\addvspace{ \l__enumext_minipage_after_skip }
2926       }
2927       { \__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.

```
2928     \bool_set_false:N \l__enumext_keyans_env_bool
2929     \__enumext_after_stop_list_v:
2930     \__enumext_vspace_below_v:
2931   }
```

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

10.35   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.
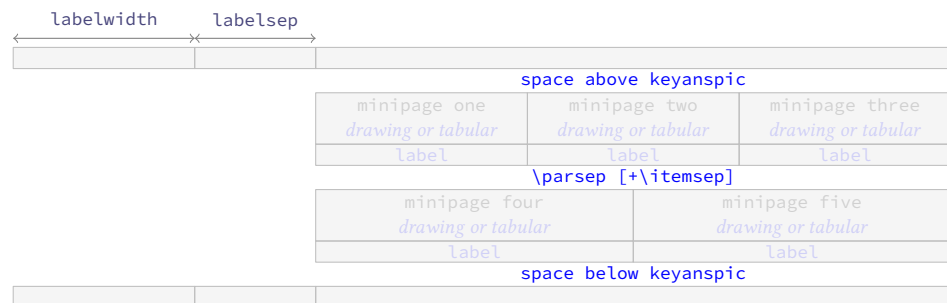
Figure 12: Representation of the keyanspic spacing in enumext.

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?.

10.35.1   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.

```
2932 \NewDocumentCommand \anspic { s o +m }
2933   {
```

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

```
2934     \bool_if:NF \l__enumext_store_active_bool
2935       {
2936         \msg_error:nnnn { enumext } { wrong-place }{ keyanspic }{ save-ans }
2937       }
2938     \int_compare:nNnT { \l__enumext_level_int } > { 1 }
2939       {
2940         \msg_error:nn { enumext } { keyanspic-wrong-level }
2941       }
```

```
2942        \int_compare:nNnT { \l__enumext_keyans_level_int } = { 1 }
2943          {
2944            \msg_error:nnnn { enumext } { command-wrong-place }{ anspic }{ keyans }
2945          }
```

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.

```
2946        \seq_put_right:Nn \l__enumext_keyans_pic_body_seq
2947          {
2948            \__enumext_keyans_anspic_code:nnn { #1 } { #2 } { #3 }
2949          }
2950      }
```

(*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.

```
2951  \cs_new_protected:Nn \__enumext_keyans_anspic_code:nnn
2952    {
2953      \stepcounter { enumXvi }
2954      #3 \\
2955      \bool_if:nT { #1 }
2956        {
2957          \__enumext_keyans_addto_prop:n { #2 }
2958          \__enumext_keyans_store_ref:
2959          \__enumext_keyans_addto_seq:n { #2 }
2960          \bool_lazy_or:nnT
2961            { \bool_if_p:N \l__enumext_show_answer_bool }
2962            { \bool_if_p:N \l__enumext_show_position_bool }
2963            {
2964              \tl_set_eq:NN \l__enumext_label_v_tl \l__enumext_label_vi_tl
2965              \__enumext_keyans_show_left:n { #2 }
2966              \tl_set_eq:NN \l__enumext_label_vi_tl \l__enumext_label_v_tl
2967            }
2968        }
2969      \tl_use:N \l__enumext_label_font_style_v_tl
2970      \__enumext_wrapper_label_v:n { \l__enumext_label_vi_tl } \__enumext_keyans_show_item_opt:
2971    }
```

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

10.35.2   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.

```
2972  \NewDocumentEnvironment{keyanspic}{ o }
2973    {
2974      \__enumext_keyans_pic_safe_exec:
2975      \__enumext_start_list:nn
2976        { }
2977        {
2978          \__enumext_keyans_pic_arg_two:
2979        }
```

We apply the "adjusted" vertical spacing above the environment

```
2980        \vspace { \l__enumext_keyans_pic_above_skip }
2981      }
```

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.

```
2982    {
2983      \tl_if_novalue:nTF { #1 }
2984        {
2985          \__enumext_keyans_pic_do:e { \seq_count:N \l__enumext_keyans_pic_body_seq }
2986        }
2987        { \__enumext_keyans_pic_do:n { #1 } }
2988      \__enumext_stop_list:
2989      \__enumext_keyans_check_ans:nn { anspic } { keyanspic }
2990      \setcounter { enumXvi } { 0 }
2991      \vspace { \l__enumext_topsep_v_skip }
```

```
2992        %\bool_set_false:N \l__enumext_store_active_bool
2993      }
```

(*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.

```
2994  \cs_new_protected:Nn \__enumext_keyans_pic_safe_exec:
2995    {
2996      \int_incr:N \l__enumext_keyans_pic_level_int
2997      \int_compare:nNnT { \l__enumext_keyans_pic_level_int } > { 1 }
2998        {
2999          \msg_error:nn { enumext } { keyanspic-nested }
3000        }
3001    }
```

(*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.

```
3002  \cs_new_protected:Npn \__enumext_keyans_pic_skip_abs:N #1
3003    {
3004      \dim_compare:nNnT { #1 } < { 0pt }
3005        { \skip_set:Nn #1 { -#1 } }
3006    }
```

(*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.

```
3007  \cs_new_protected:Nn \__enumext_keyans_pic_arg_two:
3008    {
```

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.

```
3009        \bool_set_false:N \l__enumext_leftmargin_tmp_v_bool
3010        \__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.

```
3011        \skip_add:Nn \parsep { \itemsep }
3012        \dim_add:Nn  \leftmargin { -\labelwidth - \labelsep }
3013        \dim_zero:N  \labelwidth
3014        \dim_zero:N  \listparindent
3015        \dim_zero:N  \labelsep
3016        \skip_zero:N \partopsep
3017        \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.

```
3018        \__enumext_keyans_pic_skip_abs:N \parsep
3019        \skip_set:Nn \l__enumext_keyans_pic_above_skip
3020          {
3021            \box_dp:N \strutbox
3022            + \l__enumext_topsep_v_skip
3023            - \parsep
3024          }
3025        \__enumext_item_std:w \scan_stop:
3026    }
```

(*End of definition for* __enumext_keyans_pic_arg_two:.)

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

```
3027  \cs_new_protected:Nn \__enumext_keyans_pic_do:n
3028    {
3029      \clist_map_function:nN { #1 } \__enumext_keyans_pic_row:n
3030    }
3031  \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.

```
3032 \cs_new_protected:Nn \__enumext_keyans_pic_row:n
3033   {
3034     \dim_set:Nn \l__enumext_keyans_pic_width_dim { \linewidth / #1 }
3035     \int_set:Nn \l__enumext_keyans_pic_above_int { \l__enumext_keyans_pic_below_int }
3036     \int_set:Nn \l__enumext_keyans_pic_below_int { \l__enumext_keyans_pic_above_int + #1 }
3037     \int_step_inline:nnn
3038       { \l__enumext_keyans_pic_above_int + 1 }
3039       { \l__enumext_keyans_pic_below_int }
3040       {
3041         \__enumext_minipage:w [ b ]{ \l__enumext_keyans_pic_width_dim }
3042           \centering
3043           \seq_item:Nn \l__enumext_keyans_pic_body_seq { ##1 }
3044         \__enumext_endminipage:
3045       }
3046     \par
3047   }
```

(End of definition for __enumext_keyans_pic_row:n.)

10.36 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*⟩).

10.36.1 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.

```
3048 \cs_new_protected:Nn \__enumext_starred_columns_set_vii:
3049   {
3050     \dim_compare:nNnT { \l__enumext_columns_sep_vii_dim } = { \c_zero_dim }
3051       {
3052         \dim_set:Nn \l__enumext_columns_sep_vii_dim
3053           {
3054             ( \l__enumext_labelwidth_vii_dim + \l__enumext_labelsep_vii_dim )
3055             / \l__enumext_columns_vii_int
3056           }
3057       }
3058     \int_set:Nn \l__enumext_tmpa_vii_int { \l__enumext_columns_vii_int - \c_one_int }
3059     \dim_set:Nn \l__enumext_item_width_vii_dim
3060       {
3061         ( \linewidth - \l__enumext_columns_sep_vii_dim * \l__enumext_tmpa_vii_int )
3062         / \l__enumext_columns_vii_int - \l__enumext_labelwidth_vii_dim
3063         - \l__enumext_labelsep_vii_dim
3064       }
3065     \dim_zero_new:N \itemwidth
3066   }
```

(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.

```
3067 \cs_new_protected:Npn \__enumext_starred_joined_item_vii:n #1
3068   {
3069     \int_set:Nn \l__enumext_joined_item_vii_int {#1}
3070     \int_compare:nNnT { \l__enumext_joined_item_vii_int } > { \l__enumext_columns_vii_int }
3071       {
3072         \msg_warning:nnee { enumext } { item-joined }
3073           { \int_use:N \l__enumext_joined_item_vii_int }
```

```
3074          { \int_use:N \l__enumext_columns_vii_int }
3075        \int_set:Nn \l__enumext_joined_item_vii_int
3076          {
3077            \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + \c_one_int
3078          }
3079      }
3080    \int_compare:nNnT
3081      { \l__enumext_joined_item_vii_int }
3082      >
3083      { \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + \c_one_int }
3084      {
3085        \msg_warning:nnee { enumext } { item-joined-columns }
3086          { \int_use:N \l__enumext_joined_item_vii_int }
3087          {
3088            \int_eval:n
3089              { \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + \c_one_int }
3090          }
3091        \int_set:Nn \l__enumext_joined_item_vii_int
3092          {
3093            \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + \c_one_int
3094          }
3095      }
```

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

```
3096      \int_compare:nNnTF { \l__enumext_joined_item_vii_int } > { \c_one_int }
3097        {
3098          \int_set_eq:NN \l__enumext_joined_item_aux_vii_int \l__enumext_joined_item_vii_int
3099          \int_decr:N \l__enumext_joined_item_aux_vii_int
3100          \int_add:Nn \l__enumext_item_column_pos_vii_int { \l__enumext_joined_item_aux_vii_int }
3101          \int_gadd:Nn \g__enumext_item_count_all_vii_int { \l__enumext_joined_item_aux_vii_int }
3102          \dim_set:Nn \l__enumext_joined_width_vii_dim
3103            {
3104              \l__enumext_item_width_vii_dim * \l__enumext_joined_item_vii_int
3105              + (   \l__enumext_labelwidth_vii_dim + \l__enumext_labelsep_vii_dim
3106                + \l__enumext_columns_sep_vii_dim
3107              )*\l__enumext_joined_item_aux_vii_int
3108            }
3109          \dim_set_eq:NN \itemwidth \l__enumext_joined_width_vii_dim
3110        }
3111        {
3112          \dim_set_eq:NN \l__enumext_joined_width_vii_dim \l__enumext_item_width_vii_dim
3113          \dim_set_eq:NN \itemwidth \l__enumext_item_width_vii_dim
3114        }
3115    }
```

(*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`.

```
3116  \cs_new_protected:Nn \__enumext_start_mini_vii:
3117    {
3118      \dim_compare:nNnT { \l__enumext_minipage_right_vii_dim } > { \c_zero_dim }
3119        {
3120          \dim_set:Nn \l__enumext_minipage_left_vii_dim
3121            {
3122              \linewidth
3123              - \l__enumext_minipage_right_vii_dim
3124              - \l__enumext_minipage_hsep_vii_dim
3125            }
3126          \bool_set_true:N \l__enumext_minipage_active_vii_bool
3127          \dim_gset_eq:NN
3128            \g__enumext_minipage_right_vii_dim
3129            \l__enumext_minipage_right_vii_dim
3130          \__enumext_mini_addvspace_vii:
3131          \nointerlineskip\noindent
3132          \begin{__enumext_mini_env*}{ \l__enumext_minipage_left_vii_dim }
3133        }
3134    }
```

(*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"*.

```
3135  \cs_new_protected:Nn \__enumext_stop_mini_vii:
3136    {
3137      \bool_if:NT \l__enumext_minipage_active_vii_bool
3138        {
3139          \end{__enumext_mini_env*}
3140          \hfill
3141          \bool_gset_true:N \g__enumext_minipage_active_vii_bool
3142        }
3143    }
```

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"*.

```
3144  \__enumext_after_env:nn {enumext*}
3145    {
3146      \bool_if:NT \g__enumext_minipage_active_vii_bool
3147        {
3148          \begin{__enumext_mini_env*}{ \g__enumext_minipage_right_vii_dim }
3149            \par\addvspace { \g__enumext_minipage_right_skip }
3150            \bool_if:NF \g__enumext_minipage_center_vii_bool
3151              {
3152                \centering
3153              }
3154            \tl_use:N \g__enumext_miniright_code_vii_tl % the code
3155          \end{__enumext_mini_env*}
3156          \par\addvspace{ \g__enumext_minipage_after_skip }
3157        }
3158      \bool_gset_false:N \g__enumext_minipage_active_vii_bool
3159      \bool_gset_true:N \g__enumext_minipage_center_vii_bool
3160      \tl_gclear:N \g__enumext_miniright_code_vii_tl
3161      \dim_gzero:N \g__enumext_minipage_right_vii_dim
3162      \bool_gset_false:N \g__enumext_starred_bool
3163    }
```

(*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.

```
3164  \NewDocumentEnvironment{enumext*}{ o }
3165    {
3166      \__enumext_safe_exec_vii:
3167      \__enumext_parse_keys_vii:n {#1}
3168      \__enumext_before_list_vii:
3169      \__enumext_start_store_level_vii:
3170      \__enumext_start_list:nn { }
3171        {
3172          \__enumext_list_arg_two_vii:
3173          \__enumext_before_keys_exec_vii:
3174        }
3175      \__enumext_starred_columns_set_vii:
3176      \item[] \scan_stop:
3177      \cs_set_eq:NN \__enumext_stop_item_tmp_vii: \noindent
3178      \cs_set_eq:NN \item \__enumext_start_item_tmp_vii:
3179    }
3180    {
3181      \__enumext_stop_item_tmp_vii:
3182      \__enumext_remove_extra_parsep_vii:
3183      \__enumext_stop_list:
3184      \__enumext_stop_store_level_vii:
3185      \__enumext_after_list_vii:
3186    }
```

(*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`.

```
3187 \cs_new_protected:Nn \__enumext_safe_exec_vii:
3188   {
3189     \__enumext_current_env_set_bool:
3190     \int_incr:N \l__enumext_level_h_int
3191     \int_compare:nNnT { \l__enumext_level_h_int } > { 1 }
3192       {
3193         \msg_error:nn { enumext } { nested }
3194       }
3195     \bool_set_true:N \l__enumext_starred_bool
3196     \bool_lazy_all:nT
3197       {
3198         { \bool_if_p:N \g__enumext_starred_bool }
3199         { \int_compare_p:nNn { \l__enumext_level_h_int } = { 1 } }
3200         { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
3201       }
3202       {
3203         \typeout{[[ON-FIRST-LEVEL-ENUMEXT*-NOT-NESTED]]}
3204         \bool_set_true:N \l__enumext_starred_level_one_bool
3205       }
3206   }
```

(*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.

```
3207 \cs_new_protected:Npn \__enumext_parse_keys_vii:n #1
3208   {
3209     \tl_if_novalue:nF {#1}
3210       {
3211         \str_clear:N \l__enumext_series_str
3212         \keys_set:nn { enumext / enumext* } {#1}
3213         \__enumext_parse_series:n {#1}
3214         \bool_if:NT \l__enumext_store_active_bool
3215           {
3216             \__enumext_parse_store_keys_vii:n {#1}
3217           }
3218       }
3219   }
```

(*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:`.

```
3220 \cs_new_protected:Npn \__enumext_parse_store_keys_vii:n #1
3221   {
3222     \bool_if:NF \l__enumext_store_columns_vii_bool
3223       {
3224         \regex_match:nnT { \b columns\b } {#1}
3225           {
3226             \int_set_eq:NN
3227               \l__enumext_store_columns_vii_int
3228               \l__enumext_columns_vii_int
3229             \tl_put_right:Ne \l__enumext_store_opt_vii_tl
3230               {
3231                 columns = \exp_not:V \l__enumext_store_columns_vii_int ,
3232               }
3233           }
3234       }
3235     \bool_if:NF \l__enumext_store_columns_sep_vii_bool
3236       {
3237         \regex_match:nnT { \b columns-sep \b} {#1}
3238           {
3239             \dim_set_eq:NN
3240               \l__enumext_store_columns_sep_vii_dim
```

```
3241                     \l__enumext_columns_sep_vii_dim
3242               \tl_put_right:Ne \l__enumext_store_opt_vii_tl
3243                  {
3244                    columns-sep = \exp_not:V \l__enumext_store_columns_sep_vii_dim,
3245                  }
3246             }
3247         }
3248     }
```

(*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.

```
3249 \cs_new_protected:Nn \__enumext_before_list_vii:
3250   {
3251     \__enumext_vspace_above_vii:
3252     \__enumext_check_ans_exec: % need by chek-ans
3253     \__enumext_before_args_exec_vii:
3254     \__enumext_start_mini_vii:
3255   }
```

(*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.

```
3256 \cs_new_protected:Nn \__enumext_after_list_vii:
3257   {
3258     \__enumext_stop_mini_vii:
3259     \__enumext_after_stop_list_vii:
3260     \__enumext_vspace_below_vii:
3261     \bool_set_false:N \l__enumext_starred_bool
3262     \__enumext_resume_save_counter:
3263   }
```

(*End of definition for* __enumext_after_list_vii:.)

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

```
3264 \cs_new_protected:Nn \__enumext_start_store_level_vii:
3265   {
3266     \bool_if:NT \l__enumext_store_active_bool
3267       {
3268         \int_compare:nNnT { \l__enumext_level_int } > { \c_zero_int }
3269           {
3270             \__enumext_store_level_open_vii:
3271           }
3272       }
3273   }
3274 \cs_new_protected:Nn \__enumext_stop_store_level_vii:
3275   {
3276     \bool_if:NT \l__enumext_store_active_bool
3277       {
3278         \int_compare:nNnT { \l__enumext_level_int } > { \c_zero_int }
3279           {
3280             \__enumext_store_level_close_vii:
3281           }
3282       }
3283   }
```

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

10.36.2 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`.

```
3284  \cs_new_protected_nopar:Nn \__enumext_start_item_tmp_vii:
3285    {
3286      \__enumext_stop_item_tmp_vii:
3287      \int_incr:N \l__enumext_item_column_pos_vii_int
3288      \int_gincr:N \g__enumext_item_count_all_vii_int
3289      \__enumext_item_peek_args_vii:
3290    }
```

(*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`).

```
3291  \cs_new_protected:Nn \__enumext_item_peek_args_vii:
3292    {
3293      \peek_meaning:NTF (
3294        { \__enumext_joined_item_vii:w }
3295        { \__enumext_joined_item_vii:w (1) }
3296    }
```

(*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`.

```
3297  \cs_new_protected:Npn \__enumext_joined_item_vii:w (#1)
3298    {
3299      \__enumext_starred_joined_item_vii:n {#1}
3300      \peek_meaning_remove:NTF *
3301        { \__enumext_starred_item_vii:w  }
3302        { \__enumext_standard_item_vii:w }
3303    }
```

(*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`].

```
3304  \cs_new_protected:Npn \__enumext_standard_item_vii:w
3305    {
3306      \bool_set_false:N \l__enumext_item_starred_vii_bool
3307        \peek_meaning:NTF [
3308          {
3309            \bool_set_eq:NN
3310              \l__enumext_wrap_label_vii_bool
3311              \l__enumext_wrap_label_opt_vii_bool
3312            \__enumext_start_item_vii:w
3313          }
3314          {
3315            \bool_set_true:N \l__enumext_wrap_label_vii_bool
3316            \legacy_if_set_true:n { @noitemarg }
3317            \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
3318          }
3319    }
```

(*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⟩].

```
3320 \cs_new_protected:Npn \__enumext_starred_item_vii:w
3321   {
3322     \bool_set_true:N \l__enumext_item_starred_vii_bool
3323     \bool_set_true:N \l__enumext_wrap_label_vii_bool
3324     \peek_meaning:NTF [
3325       { \__enumext_starred_item_vii_aux_i:w }
3326       { \__enumext_starred_item_vii_aux_ii:w }
3327   }
3328 \cs_new_protected:Npn \__enumext_starred_item_vii_aux_i:w [#1]
3329   {
3330     \tl_gset:Nn \g__enumext_item_symbol_aux_vii_tl {#1}
3331     \__enumext_starred_item_vii_aux_ii:w
3332   }
3333 \cs_new_protected:Npn \__enumext_starred_item_vii_aux_ii:w
3334   {
3335     \peek_meaning:NTF [
3336       { \__enumext_starred_item_vii_aux_iii:w }
3337       {
3338         \dim_set_eq:NN
3339           \l__enumext_item_symbol_sep_vii_dim
3340           \l__enumext_labelsep_vii_dim
3341         \legacy_if_set_true:n { @noitemarg }
3342         \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
3343       }
3344   }
3345 \cs_new_protected:Npn \__enumext_starred_item_vii_aux_iii:w [#1]
3346   {
3347     \dim_set:Nn \l__enumext_item_symbol_sep_vii_dim {#1}
3348     \legacy_if_set_true:n { @noitemarg }
3349     \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
3350   }
```

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

10.36.3 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.

```
3351 \cs_new_protected_nopar:Npn \__enumext_start_item_vii:w [#1]
3352   {
3353     \cs_set_eq:NN \__enumext_stop_item_tmp_vii: \__enumext_stop_item_vii:
3354     \legacy_if:nT { @noitemarg }
3355       {
3356         \legacy_if_set_false:n { @noitemarg }
3357         \legacy_if:nT { @nmbrlist }
3358           {
3359             \bool_if:NT \l__enumext_hyperref_bool
3360               {
3361                 \legacy_if_set_true:n { @hyper@item }
3362               }
3363             \refstepcounter{enumXvii}
3364             \bool_if:NT \l__enumext_check_ans_bool
3365               {
3366                 \int_gincr:N \g__enumext_count_item_number_int
3367               }
3368           }
3369       }
```

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.

```
3370      \group_begin:
3371        \lrbox{ \l__enumext_item_text_vii_box }
3372          \bool_if:NF \l__enumext_footnotes_key_bool
3373            {
3374              \__enumext_renew_footnote:
3375            }
3376          \bool_if:NT \l__enumext_item_starred_vii_bool
3377            {
3378              \tl_if_blank:VT \g__enumext_item_symbol_aux_vii_tl
3379                {
3380                  \tl_gset_eq:NN
3381                    \g__enumext_item_symbol_aux_vii_tl \l__enumext_item_symbol_vii_tl
3382                }
3383              \mode_leave_vertical:
3384              \skip_horizontal:n { -\l__enumext_item_symbol_sep_vii_dim }
3385              \makebox[ 0pt ][ r ]{ \g__enumext_item_symbol_aux_vii_tl }
3386              \skip_horizontal:N \l__enumext_item_symbol_sep_vii_dim
3387              \tl_gclear:N \g__enumext_item_symbol_aux_vii_tl
3388            }
3389          \group_begin:
3390            \tl_use:N \l__enumext_label_font_style_vii_tl
3391            \bool_if:NTF \l__enumext_wrap_label_vii_bool
3392              {
3393                \makebox[ \l__enumext_labelwidth_vii_dim ][ \l__enumext_align_label_vii_str ]
3394                  { \__enumext_wrapper_label_vii:n {#1} }
3395              }
3396              {
3397                \makebox[ \l__enumext_labelwidth_vii_dim ][ \l__enumext_align_label_vii_str ]{ #1 }
3398              }
3399          \group_end:
3400          \skip_horizontal:N \l__enumext_labelsep_vii_dim
3401          \tl_use:N \l__enumext_after_list_args_vii_tl
3402          \__enumext_minipage:w [ t ]{ \l__enumext_joined_width_vii_dim  }
3403            \skip_set_eq:NN \parindent \l__enumext_listparindent_vii_dim
3404            \skip_set_eq:NN \parskip \l__enumext_parsep_vii_skip
3405            \tl_use:N \l__enumext_fake_item_indent_vii_tl
3406      }
```

(*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.

```
3407  \cs_new_protected_nopar:Nn \__enumext_stop_item_vii:
3408    {
3409        \__enumext_endminipage:
3410      \endlrbox
3411    \group_end:
3412    \box_set_wd:Nn \l__enumext_item_text_vii_box
3413      {
3414        \l__enumext_joined_width_vii_dim
3415        + \l__enumext_labelwidth_vii_dim
3416        + \l__enumext_labelsep_vii_dim
3417      }
3418    \int_set:Nn \hbadness { 10000 }
3419    \box_use:N \l__enumext_item_text_vii_box
3420    \bool_if:NF \l__enumext_footnotes_key_bool
3421      {
3422        \__enumext_print_footnote:
3423      }
3424    \int_compare:nNnTF { \l__enumext_item_column_pos_vii_int } = { \l__enumext_columns_vii_int }
3425      {
3426        \par\noindent
3427        \int_zero:N \l__enumext_item_column_pos_vii_int
3428      }
3429      { \hspace{ \l__enumext_columns_sep_vii_dim } }
3430    }
```

(*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.

```
3431  \cs_new_protected:Nn \__enumext_remove_extra_parsep_vii:
3432    {
3433      \int_compare:nNnT
3434        {
3435          \int_mod:nn {  \g__enumext_item_count_all_vii_int } { \l__enumext_columns_vii_int }
3436        }
3437        =
3438        { \c_zero_int }
3439        {
3440          \par
3441          \vspace{ -\l__enumext_itemsep_vii_skip }
3442          \int_gzero:N \g__enumext_item_count_all_vii_int
3443        }
3444    }
```

(*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.

```
3445  \__enumext_after_env:nn {enumext*}
3446    {
3447      \int_compare:nNnT { \l__enumext_level_int } = { 0 }
3448        {
3449          \bool_if:NT \g__enumext_check_ans_show_h_bool
3450            {
3451              \__enumext_check_ans_show:
3452            }
3453          \bool_gset_false:N \g__enumext_starred_bool
3454          \bool_gset_false:N \g__enumext_check_ans_show_h_bool
3455          \tl_gclear:N \g__enumext_store_name_tl
3456        }
3457    }
```

10.37 The environment keyans*

10.37.1 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.

```
3458  \cs_new_protected:Nn \__enumext_starred_columns_set_viii:
3459    {
3460      \dim_compare:nNnT { \l__enumext_columns_sep_viii_dim } = { \c_zero_dim }
3461        {
3462          \dim_set:Nn \l__enumext_columns_sep_viii_dim
3463            {
3464              ( \l__enumext_labelwidth_viii_dim + \l__enumext_labelsep_viii_dim )
3465              / \l__enumext_columns_viii_int
3466            }
3467        }
3468      \int_set:Nn \l__enumext_tmpa_viii_int { \l__enumext_columns_viii_int - \c_one_int }
3469      \dim_set:Nn \l__enumext_item_width_viii_dim
3470        {
3471          ( \linewidth - \l__enumext_columns_sep_viii_dim * \l__enumext_tmpa_viii_int )
3472          / \l__enumext_columns_viii_int - \l__enumext_labelwidth_viii_dim
3473          - \l__enumext_labelsep_viii_dim
3474        }
3475      \dim_zero_new:N \itemwidth
3476    }
```

(*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.

```
3477  \cs_new_protected:Npn \__enumext_starred_joined_item_viii:n #1
3478    {
3479      \int_set:Nn \l__enumext_joined_item_viii_int {#1}
3480      \int_compare:nNnT { \l__enumext_joined_item_viii_int } > { \l__enumext_columns_viii_int }
3481        {
3482          \msg_warning:nnee { enumext } { item-joined }
3483            { \int_use:N \l__enumext_joined_item_viii_int }
```

```
3484                { \int_use:N \l__enumext_columns_viii_int }
3485            \int_set:Nn \l__enumext_joined_item_viii_int
3486              {
3487                \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + \c_one_int
3488              }
3489          }
3490        \int_compare:nNnT
3491          { \l__enumext_joined_item_viii_int }
3492            >
3493          { \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + \c_one_int }
3494          {
3495            \msg_warning:nnee { enumext } { item-joined-columns }
3496              { \int_use:N \l__enumext_joined_item_viii_int }
3497              {
3498                \int_eval:n
3499                  { \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + \c_one_int }
3500              }
3501            \int_set:Nn \l__enumext_joined_item_viii_int
3502              {
3503                \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + \c_one_int
3504              }
3505          }
3506        \int_compare:nNnTF { \l__enumext_joined_item_viii_int } > { \c_one_int }
3507          {
3508            \int_set_eq:NN \l__enumext_joined_item_aux_viii_int \l__enumext_joined_item_viii_int
3509            \int_decr:N \l__enumext_joined_item_aux_viii_int
3510            \int_add:Nn \l__enumext_item_column_pos_viii_int { \l__enumext_joined_item_aux_viii_int }
3511            \int_gadd:Nn \g__enumext_item_count_all_viii_int { \l__enumext_joined_item_aux_viii_int }
3512            \dim_set:Nn \l__enumext_joined_width_viii_dim
3513              {
3514                \l__enumext_item_width_viii_dim * \l__enumext_joined_item_viii_int
3515                + (  \l__enumext_labelwidth_viii_dim + \l__enumext_labelsep_viii_dim
3516                  + \l__enumext_columns_sep_viii_dim
3517                )*\l__enumext_joined_item_aux_viii_int
3518              }
3519            \dim_set_eq:NN \itemwidth \l__enumext_joined_width_viii_dim
3520          }
3521          {
3522            \dim_set_eq:NN \l__enumext_joined_width_viii_dim \l__enumext_item_width_viii_dim
3523            \dim_set_eq:NN \itemwidth \l__enumext_item_width_viii_dim
3524          }
3525      }
```

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

__enumext_start_mini_viii:
__enumext_stop_mini_viii:

The implementation of the `mini-env` key is identical to the one used in the `enumext*` environment.

```
3526  \cs_new_protected:Nn \__enumext_start_mini_viii:
3527    {
3528      \dim_compare:nNnT { \l__enumext_minipage_right_viii_dim } > { \c_zero_dim }
3529        {
3530          \dim_set:Nn \l__enumext_minipage_left_viii_dim
3531            {
3532              \linewidth
3533              - \l__enumext_minipage_right_viii_dim
3534              - \l__enumext_minipage_hsep_viii_dim
3535            }
3536          \bool_set_true:N \l__enumext_minipage_active_viii_bool
3537          \dim_gset_eq:NN
3538            \g__enumext_minipage_right_viii_dim
3539            \l__enumext_minipage_right_viii_dim
3540          \__enumext_mini_addvspace_viii:
3541          \nointerlineskip\noindent
3542          \begin{__enumext_mini_env*}{ \l__enumext_minipage_left_viii_dim }
3543        }
3544    }
3545  \cs_new_protected:Nn \__enumext_stop_mini_viii:
3546    {
3547      \bool_if:NT \l__enumext_minipage_active_viii_bool
3548        {
3549          \end{__enumext_mini_env*}
3550          \hfill
```

```
3551            \bool_gset_true:N \g__enumext_minipage_active_viii_bool
3552          }
3553      }
3554  \__enumext_after_env:nn {keyans*}
3555    {
3556      \bool_if:NT \g__enumext_minipage_active_viii_bool
3557        {
3558          \begin{__enumext_mini_env*}{ \g__enumext_minipage_right_viii_dim }
3559            \par\addvspace { \g__enumext_minipage_right_skip }
3560            \bool_if:NF \g__enumext_minipage_center_viii_bool
3561              {
3562                \centering
3563              }
3564            \tl_use:N \g__enumext_miniright_code_viii_tl % the code
3565          \end{__enumext_mini_env*}
3566          \par\addvspace{ \g__enumext_minipage_after_skip }
3567        }
3568      \bool_gset_false:N \g__enumext_minipage_active_viii_bool
3569      \bool_gset_true:N \g__enumext_minipage_center_viii_bool
3570      \tl_gclear:N \g__enumext_miniright_code_viii_tl
3571      \dim_gzero:N \g__enumext_minipage_right_viii_dim
3572    }
```

(*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.

```
3573  \NewDocumentEnvironment{keyans*}{ o }
3574    {
3575      \__enumext_safe_exec_viii:
3576      \__enumext_parse_keys_viii:n {#1}
3577      \__enumext_before_list_viii:
3578      \__enumext_start_list:nn { }
3579        {
3580          \__enumext_list_arg_two_viii:
3581          \__enumext_before_keys_exec_viii:
3582        }
3583      \__enumext_starred_columns_set_viii:
3584      \item[] \scan_stop:
3585      \cs_set_eq:NN \__enumext_stop_item_tmp_viii: \noindent
3586      \cs_set_eq:NN \item \__enumext_start_item_tmp_viii:
3587    }
3588    {
3589      \__enumext_stop_item_tmp_viii:
3590      \__enumext_remove_extra_parsep_viii:
3591      \__enumext_keyans_check_ans:nn { item }{ keyans* }
3592      \__enumext_stop_list:
3593      \__enumext_after_list_viii:
3594    }
```

(*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.

```
3595  \cs_new_protected:Nn \__enumext_safe_exec_viii:
3596    {
3597      \int_incr:N \l__enumext_keyans_level_h_int
3598      \int_compare:nNnT { \l__enumext_keyans_level_h_int } > { 1 }
3599        {
3600          \msg_error:nn { enumext } { nested }
3601        }
3602    % Set false for interfering with enumext nested in keyans* (yes, its possible and crayze)
3603      \bool_set_false:N \l__enumext_store_active_bool
3604      \int_compare:nNnT { \l__enumext_level_int } > { 1 }
3605        {
3606          \msg_error:nn { enumext } { keyans-wrong-level }
3607        }
3608    }
```

(*End of definition for* __enumext_safe_exec_viii:.)

__enumext_parse_keys_viii:n

Parse [⟨*key* = *val*⟩] for keyans*.

```
3609 \cs_new_protected:Npn \__enumext_parse_keys_viii:n #1
3610   {
3611     \tl_if_novalue:nF {#1}
3612       {
3613         \keys_set:nn { enumext / keyans* } {#1}
3614       }
3615   }
```

(*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.

```
3616 \cs_new_protected:Nn \__enumext_before_list_viii:
3617   {
3618     \__enumext_vspace_above_viii:
3619     \__enumext_before_args_exec_viii:
3620     \__enumext_start_mini_viii:
3621   }
```

(*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.

```
3622 \cs_new_protected:Nn \__enumext_after_list_viii:
3623   {
3624     \__enumext_stop_mini_viii:
3625     \__enumext_after_stop_list_viii:
3626     \__enumext_vspace_below_viii:
3627   }
```

(*End of definition for* __enumext_after_list_viii:.)

10.37.2 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.

```
3628 \cs_new_protected_nopar:Nn \__enumext_start_item_tmp_viii:
3629   {
3630     \__enumext_stop_item_tmp_viii:
3631     \int_incr:N \l__enumext_item_column_pos_viii_int
3632     \int_gincr:N \g__enumext_item_count_all_viii_int
3633     \__enumext_item_peek_args_viii:
3634   }
```

(*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).

```
3635 \cs_new_protected:Nn \__enumext_item_peek_args_viii:
3636   {
3637     \peek_meaning:NTF (
3638       { \__enumext_joined_item_viii:w }
3639       { \__enumext_joined_item_viii:w (1) }
3640   }
```

(*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.

```
3641 \cs_new_protected:Npn \__enumext_joined_item_viii:w (#1)
3642   {
3643     \__enumext_starred_joined_item_viii:n {#1}
3644     \peek_meaning_remove:NTF *
3645       { \__enumext_starred_item_viii:w  }
3646       { \__enumext_standard_item_viii:w }
3647   }
```

(*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].

```
3648 \cs_new_protected:Npn \__enumext_standard_item_viii:w
3649   {
3650     \bool_set_false:N \l__enumext_item_starred_viii_bool
3651     \peek_meaning:NTF [
3652       {
3653         \bool_set_eq:NN
3654           \l__enumext_wrap_label_viii_bool
3655           \l__enumext_wrap_label_opt_viii_bool
3656         \__enumext_start_item_viii:w
3657       }
3658       {
3659         \bool_set_true:N \l__enumext_wrap_label_viii_bool
3660         \legacy_if_set_true:n { @noitemarg }
3661         \__enumext_start_item_viii:w [ \l__enumext_label_viii_tl ]
3662       }
3663   }
```

(*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*⟩].

```
3664 \cs_new_protected:Npn \__enumext_starred_item_viii:w
3665   {
3666     \bool_set_true:N \l__enumext_item_starred_viii_bool
3667     \bool_set_true:N \l__enumext_wrap_label_viii_bool
3668     \peek_meaning:NTF [
3669       { \__enumext_starred_item_viii_aux_i:w }
3670       { \__enumext_starred_item_viii_aux_ii:w }
3671   }
```

The optional argument will be captured in the variables \l__enumext_keyans_tmpa_tl and \l__enumext_keyans_tmpb_tl which we will use later for the implementation of the show-ans and show-pos keys together with the stored in ⟨*sequence*⟩ and ⟨*prop list*⟩.

```
3672 \cs_new_protected:Npn \__enumext_starred_item_viii_aux_i:w [#1]
3673   {
3674     \tl_clear:N \l__enumext_store_keyans_label_tl
3675     \tl_if_novalue:nF { #1 }
3676       {
3677         \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_tl
3678           {
3679             \tl_put_right:Ne \l__enumext_store_keyans_label_tl { \l__enumext_store_keyans_item_op⌉
3680             \tl_put_right:Ne \l__enumext_store_keyans_label_tl { #1 }
3681           }
3682         \tl_set:Ne \l__enumext_keyans_item_opt_tl { #1 }
3683       }
3684     \__enumext_starred_item_viii_aux_ii:w
3685   }
3686 \cs_new_protected:Npn \__enumext_starred_item_viii_aux_ii:w
```

```
3687    {
3688      \legacy_if_set_true:n { @noitemarg }
3689      \__enumext_start_item_viii:w [ \l__enumext_label_viii_tl ]
3690    }
```

(*End of definition for* __enumext_starred_item_viii:w, __enumext_starred_item_viii_aux_i:w, *and* __enumext_-*starred_item_viii_aux_ii:w.*)

__enumext_starred_item_exec: The function __enumext_starred_item_exec: will be in charge of storing the current ⟨*label*⟩ for \item* followed by the [⟨*content*⟩] for \item*[⟨*content*⟩] if present in the ⟨*sequence*⟩ and ⟨*prop list*⟩ set by the save-ans key. In this same function the keys show-ans, show-pos and save-ref are implemented.

```
3691  \cs_new_protected:Nn \__enumext_starred_item_exec:
3692    {
3693      \tl_put_left:Ne \l__enumext_store_keyans_label_tl { \l__enumext_label_viii_tl }
3694      \__enumext_store_addto_prop:V \l__enumext_store_keyans_label_tl
3695      \__enumext_keyans_store_ref:
3696      \tl_put_left:Ne \l__enumext_store_keyans_label_tl { \item }
3697      \__enumext_keyans_addto_seq_link:
3698      \bool_if:NT \l__enumext_show_answer_bool
3699        {
3700          \__enumext_print_keyans_box:NN \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
3701        }
3702      \bool_if:NT \l__enumext_show_position_bool
3703        {
3704          \tl_set:Ne \l__enumext_mark_answer_sym_tl
3705            {
3706              \group_begin:
3707                \exp_not:N \normalfont
3708                \exp_not:N \footnotesize [ \int_eval:n
3709                  {
3710                    \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
3711                  }
3712                ]
3713              \group_end:
3714            }
3715          \__enumext_print_keyans_box:NN \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
3716        }
3717    }
```

(*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.

```
3718  \cs_new_protected_nopar:Npn \__enumext_start_item_viii:w [#1]
3719    {
3720      \cs_set_eq:NN \__enumext_stop_item_tmp_viii: \__enumext_stop_item_viii:
3721      \legacy_if:nT { @noitemarg }
3722        {
3723          \legacy_if_set_false:n { @noitemarg }
3724          \legacy_if:nT { @nmbrlist }
3725            {
3726              \bool_if:NT \l__enumext_hyperref_bool
3727                {
3728                  \legacy_if_set_true:n { @hyper@item }
3729                }
3730              \refstepcounter{enumXviii}
3731            }
3732        }
```

Here we start capturing \item and its contents into a group using the plain form of the lrbox environment.

```
3733      \group_begin:
3734        \lrbox{ \l__enumext_item_text_viii_box }
3735          \bool_if:NF \l__enumext_footnotes_key_bool
3736            {
3737              \__enumext_renew_footnote:
3738            }
3739          \bool_if:NT \l__enumext_item_starred_viii_bool
3740            {
3741              \__enumext_starred_item_exec:
3742            }
```

```
3743          \group_begin:
3744            \tl_use:N \l__enumext_label_font_style_viii_tl
3745            \bool_if:NTF \l__enumext_wrap_label_viii_bool
3746              {
3747                \makebox[ \l__enumext_labelwidth_viii_dim ][ \l__enumext_align_label_viii_str ]
3748                  { \__enumext_wrapper_label_viii:n {#1} }
3749              }
3750              {
3751                \makebox[ \l__enumext_labelwidth_viii_dim ][ \l__enumext_align_label_viii_str ]{ #1
3752              }
3753          \group_end:
3754          \skip_horizontal:N \l__enumext_labelsep_viii_dim
3755          \tl_use:N \l__enumext_after_list_args_viii_tl
3756          \__enumext_minipage:w [ t ]{ \l__enumext_joined_width_viii_dim  }
3757            \skip_set_eq:NN \parindent \l__enumext_listparindent_viii_dim
3758            \skip_set_eq:NN \parskip \l__enumext_parsep_viii_skip
3759            \bool_if:NT \l__enumext_item_starred_viii_bool
3760              {
3761                \tl_use:N \l__enumext_fake_item_indent_viii_tl
3762                \__enumext_keyans_show_item_opt: \skip_horizontal:n { -\l__enumext_fake_item_indent_
3763              }
3764              {
3765                \tl_use:N \l__enumext_fake_item_indent_viii_tl
3766              }
3767      }
```

(*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.

```
3768  \cs_new_protected_nopar:Nn \__enumext_stop_item_viii:
3769    {
3770          \__enumext_endminipage:
3771        \endlrbox
3772      \group_end:
3773      \box_set_wd:Nn \l__enumext_item_text_viii_box
3774        {
3775          \l__enumext_joined_width_viii_dim
3776          + \l__enumext_labelwidth_viii_dim
3777          + \l__enumext_labelsep_viii_dim
3778        }
3779      \int_set:Nn \hbadness { 10000 }
3780      \box_use:N \l__enumext_item_text_viii_box
3781      \bool_if:NF \l__enumext_footnotes_key_bool
3782        {
3783          \__enumext_print_footnote:
3784        }
3785      \int_compare:nNnTF { \l__enumext_item_column_pos_viii_int } = { \l__enumext_columns_viii_int }
3786        {
3787          \par\noindent
3788          \int_zero:N \l__enumext_item_column_pos_viii_int
3789        }
3790        { \hspace{ \l__enumext_columns_sep_viii_dim } }
3791    }
```

(*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.

```
3792  \cs_new_protected:Nn \__enumext_remove_extra_parsep_viii:
3793    {
3794      \int_compare:nNnT
3795        {
3796          \int_mod:nn {  \g__enumext_item_count_all_viii_int } { \l__enumext_columns_viii_int }
3797        }
3798        =
3799        { \c_zero_int }
3800        {
3801          \par
3802          \vspace{ -\l__enumext_itemsep_viii_skip }
```

```
3803                \int_gzero:N \g__enumext_item_count_all_viii_int
3804            }
3805        }
```

(*End of definition for* __enumext_remove_extra_parsep_viii:.)

10.38   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.

```
3806   \NewDocumentCommand \getkeyans { m }
3807     {
3808        \exp_args:Ne \__enumext_getkeyans_aux:n
3809          { \tl_to_str:e { \text_expand:n {#1} } }
3810     }
```

(*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.

```
3811   \cs_new_protected:Npn \__enumext_getkeyans_aux:n #1
3812     {
3813        \str_if_in:nnTF {#1} { : }
3814          {
3815            \use:e
3816              {
3817                \cs_set:Npn \exp_not:N \__enumext_tmp:w ##1 \c_colon_str ##2 \scan_stop:
3818                  { {##1} {##2} }
3819              }
3820            \exp_after:wN \__enumext_getkeyans:nn \__enumext_tmp:w #1 \scan_stop:
3821          }
3822          { \msg_error:nnn { enumext } { missing-colon } {#1} }
3823     }
```

(*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*⟩.

```
3824   \cs_new_protected:Npn \__enumext_getkeyans:nn #1 #2
3825     {
3826        \prop_if_exist:cF { g__enumext_#1_prop }
3827          { \msg_error:nnn { enumext } { undefined-storage-anskey } {#1} }
3828        \group_begin:
3829          \prop_item:cn { g__enumext_#1_prop }{#2}
3830        \group_end:
3831     }
```

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

10.39   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.

```
3832   \keys_define:nn  { keyanskey / print }
3833     {
3834        level-1 .code:n     = \tl_put_right:Nn \l__enumext_print_keyans_i_tl
3835                               {
3836                                 \setenumext[level,1] {#1} \setenumext[print,1] {#1}
3837                               },
3838        level-1 .initial:n  = { label=\arabic*., nosep, columns=2, first=\small, font=\small },
3839        level-2 .code:n     = \tl_put_right:Nn \l__enumext_print_keyans_ii_tl
3840                               {
3841                                 \setenumext[level,2] {#1} \setenumext[print,2] {#1}
3842                               },
3843        level-2 .initial:n  = { nosep, label=(\alph*), first=\small, font=\small },
3844        level-3 .code:n     = \tl_put_right:Nn \l__enumext_print_keyans_iii_tl
3845                               {
3846                                 \setenumext[level,3] {#1} \setenumext[print,3] {#1}
3847                               },
```

```
3848      level-3 .initial:n  = { nosep, label=\roman*., first=\small, font=\small },
3849      level-4 .code:n     = \tl_put_right:Nn \l__enumext_print_keyans_iv_tl
3850                              {
3851                                \setenumext[level,4] {#1} \setenumext[print,4] {#1}
3852                              },
3853      level-4 .initial:n  = { nosep, label=\Alph*., first=\small, font=\small },
3854      level-* .code:n     = \tl_put_right:Nn \l__enumext_print_keyans_vii_tl % starred
3855                              {
3856                                \setenumext[enumext*] {#1} %%\setenumext[print,*] {#1}
3857                              },
3858      level-* .initial:n  = { label=\arabic*., nosep, columns=2, first=\small, font=\small },
3859    }
```

\printkeyans Create a user command to print *"all stored content"* in ⟨*sequence*⟩ for \anskey, \item* and \anspic*.

```
3860  \NewDocumentCommand \printkeyans { s O{} m }
3861    {
3862      \group_begin:
3863        \tl_use:N \l__enumext_print_keyans_i_tl
3864        \tl_use:N \l__enumext_print_keyans_ii_tl
3865        \tl_use:N \l__enumext_print_keyans_iii_tl
3866        \tl_use:N \l__enumext_print_keyans_iv_tl
3867        \tl_use:N \l__enumext_print_keyans_vii_tl
3868        \__enumext_printkeyans:nnn { #1 } { #2 } { #3 }
3869      \group_end:
3870    }
```

(*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

```
3871  \cs_new_protected:Npn \__enumext_printkeyans:nnn #1 #2 #3
3872    {
3873      \seq_if_exist:cTF { g__enumext_#3_seq }
3874        {
3875          \seq_if_empty:cF { g__enumext_#3_seq }
3876            {
3877              %%\seq_show:c { g__enumext_#3_seq }
3878              \bool_if:nTF {#1}
3879                {
3880                  \begin{enumext*}[#2]
3881                    \seq_map_inline:cn { g__enumext_#3_seq } { ##1 }
3882                  \end{enumext*}
3883                }
3884                {
3885                  \begin{enumext}[#2]
3886                    \seq_map_inline:cn { g__enumext_#3_seq } { ##1 }
3887                  \end{enumext}
3888                }
3889            }
3890        }
3891        {
3892          \msg_error:nnn { enumext } { undefined-storage-anskey } {#3}
3893        }
3894    }
```

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

10.40 The command \setenumext

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

```
3895  \keys_define:nn { enumext / meta-families }
3896    {
3897      level-1  .code:n = { \keys_set:nn { enumext / level-1  } {#1} } ,
3898      level-2  .code:n = { \keys_set:nn { enumext / level-2  } {#1} } ,
3899      level-3  .code:n = { \keys_set:nn { enumext / level-3  } {#1} } ,
3900      level-4  .code:n = { \keys_set:nn { enumext / level-4  } {#1} } ,
3901      keyans   .code:n = { \keys_set:nn { enumext / keyans   } {#1} } ,
```

```
3902      enumext* .code:n = { \keys_set:nn { enumext / enumext* } {#1} } ,
3903      keyans*  .code:n = { \keys_set:nn { enumext / keyans*  } {#1} } ,
3904       print-1 .code:n = { \keys_set:nn { keyanskey / print } { level-1 = {#1} } } ,
3905       print-2 .code:n = { \keys_set:nn { keyanskey / print } { level-2 = {#1} } } ,
3906       print-3 .code:n = { \keys_set:nn { keyanskey / print } { level-3 = {#1} } } ,
3907       print-4 .code:n = { \keys_set:nn { keyanskey / print } { level-4 = {#1} } } ,
3908      print-*  .code:n = { \keys_set:nn { keyanskey / print } { level-* = {#1} } } ,
3909      unknown  .code:n = { \msg_error:nn { enumext } { unknown-key-family } } ,
3910    }
```

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

```
3911  \seq_const_from_clist:Nn \c__enumext_all_families_seq
3912    {
3913      level-1 , level-2 , level-3 , level-4 , keyans, enumext*,
3914      keyans* , print-1 , print-2 , print-3 , print-4 , print-*,
3915    }
```

\setenumext   Now we define the user command \setenumext.

```
3916  \NewDocumentCommand \setenumext { o +m }
3917    {
3918      \tl_if_novalue:nTF {#1}
3919        {
3920          \seq_map_inline:Nn \c__enumext_all_families_seq
3921        }
3922        {
3923          \seq_clear:N \l__enumext_setkey_tmpa_seq
3924          \seq_set_from_clist:Nn \l__enumext_setkey_tmpb_seq {#1}
3925          \int_set:Nn \l__enumext_setkey_tmpa_int
3926            {
3927              \seq_count:N \l__enumext_setkey_tmpb_seq
3928            }
3929          \int_compare:nNnTF { \l__enumext_setkey_tmpa_int } > { 1 }
3930            {
3931              \seq_pop_left:NN \l__enumext_setkey_tmpb_seq \l__enumext_setkey_tmpa_tl
3932              \seq_map_function:NN \l__enumext_setkey_tmpb_seq \__enumext_set_parse:n
3933              \seq_set_map_e:NNn \l__enumext_setkey_tmpa_seq \l__enumext_setkey_tmpa_seq
3934                {
3935                  \tl_use:N \l__enumext_setkey_tmpa_tl - ##1
3936                }
3937            }
3938            {
3939              \seq_put_right:Ne \l__enumext_setkey_tmpa_seq { \tl_trim_spaces:n {#1} }
3940            }
3941          \seq_if_empty:NTF \l__enumext_setkey_tmpa_seq
3942            { \seq_map_inline:Nn \c__enumext_all_families_seq }
3943            { \seq_map_inline:Nn \l__enumext_setkey_tmpa_seq }
3944        }
3945        {
3946          \keys_set:nn { enumext / meta-families } { ##1 = {#2} }
3947        }
3948    }
```

(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

```
3949  \cs_new_protected:Npn \__enumext_set_parse:n #1
3950    {
3951      \tl_set:Ne \l__enumext_setkey_tmpb_tl { \tl_trim_spaces:n {#1} }
3952      \int_step_inline:nnn { 0 } { 4 } % <- max level
3953        { \tl_remove_all:Nn \l__enumext_setkey_tmpb_tl {##1} }
3954      \tl_if_empty:NTF \l__enumext_setkey_tmpb_tl
3955        {
3956          \seq_put_right:Ne \l__enumext_setkey_tmpa_seq
3957            { \tl_trim_spaces:n {#1} }
3958        }
3959        { \__enumext_set_error:nn {#1} { } }
3960    }
3961  \cs_new_protected:Npn \__enumext_set_error:nn #1 #2
3962    { \msg_error:nnn { enumext } { invalid-key } {#1} {#2} }
```

(End of definition for __enumext_set_parse:n and __enumext_set_error:nn.)

10.41   Messages

Message used by package-load for multicol and hyperref packages.

```
3963 \msg_new:nnn { enumext } { package-load }
3964   {
3965     The ~ '#1' ~ package ~ is ~ already ~ loaded.
3966   }

3967 \msg_new:nnn { enumext } { package-not-load }
3968   {
3969     The ~ '#1' ~ package ~ will ~ be ~ loaded ~ as ~ a ~ dependency.
3970   }

3971 \msg_new:nnn { enumext } { package-load-foot }
3972   {
3973     The ~ '#1' ~ package ~ is ~ loaded ~ with ~ the ~ option ~ '#2'.
3974   }
```

Message used in the creation of counters by enumext package.

```
3975 \msg_new:nnn { enumext } { counters }
3976   {
3977     The ~ counter ~ '#1' ~ is ~ already ~ defined ~ by ~ some ~ \\
3978     package ~ or ~ macro, ~ it ~ cannot ~ be ~ continued.
3979   }
```

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

```
3980 \msg_new:nnn { enumext } { invalid-key }
3981   {
3982     The ~ key ~ '#1' ~ is ~ not ~ know ~ the ~ level ~ #2.
3983   }
3984 \msg_new:nnn { enumext } { unknown-key-family }
3985   {
3986     Unknown~key~family~`\l_keys_key_str'~for~enumext.
3987   }
```

Messages used in length calculation.

```
3988 \msg_new:nnn { enumext } { width-negative }
3989   {
3990     Ignoring ~ negative ~ value ~ '#1=#2' ~ \msg_line_context:.\\
3991     The ~ key ~ '#1'~ accepts ~ values  ~ >= ~ 0pt.
3992   }
3993 \msg_new:nnn { enumext } { width-zero }
3994   {
3995     Invalid ~ '#1=#2' ~ \msg_line_context:.\\
3996     The ~ key ~ '#1'~ accepts ~ values  ~ > ~ 0pt.
3997   }
```

Messages used by show-length key in enumext.

```
3998 \msg_new:nnn { enumext } { list-lengths }
3999   {
4000     **** ~ Lengths ~ used ~ by ~ 'enumext' ~ level ~ '#2' ~ \msg_line_context:~\c_space_tl ****\\
4001     \__enumext_show_length:nnn { dim  } { labelsep    } {#1}
4002     \__enumext_show_length:nnn { dim  } { labelwidth  } {#1}
4003     \__enumext_show_length:nnn { dim  } { itemindent  } {#1}
4004     \__enumext_show_length:nnn { dim  } { leftmargin  } {#1}
4005     \__enumext_show_length:nnn { dim  } { rightmargin } {#1}
4006     \__enumext_show_length:nnn { dim  } { listparindent } {#1}
4007     \__enumext_show_length:nnn { skip } { topsep     } {#1}
4008     \__enumext_show_length:nnn { skip } { parsep     } {#1}
4009     \__enumext_show_length:nnn { skip } { partopsep  } {#1}
4010     \__enumext_show_length:nnn { skip } { itemsep    } {#1}
4011     ************************************************
4012   }
```

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

```
4013 \msg_new:nnn { enumext } { list-lengths-not-nested }
4014   {
4015     **** ~ Lengths ~ used ~ by ~ '#2' ~ environment ~ \msg_line_context:~\c_space_tl ****\\
4016     \__enumext_show_length:nnn { dim  } { labelsep    } {#1}
4017     \__enumext_show_length:nnn { dim  } { labelwidth  } {#1}
4018     \__enumext_show_length:nnn { dim  } { itemindent  } {#1}
4019     \__enumext_show_length:nnn { dim  } { leftmargin  } {#1}
4020     \__enumext_show_length:nnn { dim  } { rightmargin } {#1}
4021     \__enumext_show_length:nnn { dim  } { listparindent } {#1}
```

```
4022     \__enumext_show_length:nnn { skip } { topsep     } {#1}
4023     \__enumext_show_length:nnn { skip } { parsep     } {#1}
4024     \__enumext_show_length:nnn { skip } { partopsep } {#1}
4025     \__enumext_show_length:nnn { skip } { itemsep    } {#1}
4026     **************************************************
4027   }
```

Messages used by save-ans key.

```
4028 \msg_new:nnn { enumext } { save-ans-empty }
4029   {
4030     Key ~ 'save-ans' ~ need ~ a ~ value ~ in ~ '#1' ~ on ~ line ~ #2.
4031   }
4032 \msg_new:nnn { enumext } { save-ans-ok }
4033   {
4034     Set ~ 'save-ans=#2' ~ in ~'#1' ~ on ~ line ~ #3.
4035   }
```

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

```
4036 \msg_new:nnn { enumext } { items-same-answer }
4037   {
4038     **********~Checking~answers~on~'#1'~OK~**********\\
4039     **~ All ~ items ~ stored ~ in ~ sequence ~ '#1' ~ have ~ an ~ answer. \\
4040     ****************************************
4041     \prg_replicate:nn { 7 + \str_count:n {#1} } { * }
4042   }
4043 \msg_new:nnn { enumext } { item-different-answer }
4044   {
4045     Number ~ of ~ items ~ different ~  of ~ number ~ of ~
4046     answer ~ in ~ sequence ~ '#1'~ closed ~ \msg_line_context:.
4047   }
```

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

```
4048 \msg_new:nnn { enumext } { missing-starred }
4049   {
4050     Missing ~ '\c_backslash_str #1*' ~ in ~ '#2' ~ \msg_line_context:.
4051   }
```

Message for the nesting depth of the environment enumext.

```
4052 \msg_new:nnn { enumext } { list-too-deep }
4053   {
4054     Too ~ deep ~ nesting  ~ for ~ 'enumext' ~ \msg_line_context:.~ \\
4055     The ~ maximum  ~ level  ~ of  ~ nesting  ~ is ~ 4.
4056   }
```

Messages used by \anskey and \anspic commands.

```
4057 \msg_new:nnn { enumext } { anskey-wrong-place }
4058   {
4059     Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:.~ \\
4060     '\c_backslash_str #1' ~ works ~ in ~ the ~ environment ~ '#2'.
4061   }
4062 \msg_new:nnn { enumext } { anspic-wrong-place }
4063   {
4064     Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:.~ \\
4065     '\c_backslash_str #1' ~ works ~ in ~ the ~ environment ~ '#2'.
4066   }
4067 \msg_new:nnn { enumext } { command-wrong-place }
4068   {
4069     Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:.~ \\
4070     '\c_backslash_str #1' ~ works ~ outside ~ the ~ environment ~ '#2'.
4071   }
```

Messages used by keyans and keyanspic environment.

```
4072 \msg_new:nnn { enumext } { keyans-nested }
4073   {
4074     The ~ environment ~ 'keyans' ~ can't ~ be  ~ nested  ~ \msg_line_context:.
4075   }
4076 \msg_new:nnn { enumext } { keyans-wrong-level }
4077   {
4078     Wrong ~ level ~ position ~ for ~ 'keyans' ~ \msg_line_context:.~ \\
4079     The ~ environment ~ 'keyans' ~ can ~ only ~ be ~ in ~ the ~ first ~ level.
4080   }
4081 \msg_new:nnn { enumext } { wrong-place }
4082   {
4083     Wrong ~ place ~ for ~ '#1' ~ environment ~\msg_line_context:.~ \\
```

```
4084      '#1' ~ is ~ only ~ found ~ with ~ '#2' ~  in  ~  'enumext.
4085    }
4086 \msg_new:nnn { enumext } { keyanspic-nested }
4087    {
4088      The ~ environment ~ 'keyanspic' ~ can't ~ be  ~ nested~ \msg_line_context:.~.
4089    }
4090 \msg_new:nnn { enumext } { keyanspic-wrong-level }
4091    {
4092      Wrong ~ level ~ position ~ for ~ 'keyanspic' ~ \msg_line_context:.~ \\
4093      The ~ environment ~ 'keyans' ~ can ~ only ~ be ~ in ~ the ~ first ~ level.
4094    }
```

Messages used by \getkeyans command.

```
4095 \msg_new:nnn { enumext } { undefined-storage-anskey }
4096    {
4097      Storage ~ named ~ '#1' ~ is ~ not ~ defined ~ \msg_line_context:.
4098    }
```

Messages used by \miniright command.

```
4099 \msg_new:nnn { enumext } { missing-miniright }
4100    {
4101      Missing ~ '\c_backslash_str miniright' ~ in ~ \msg_line_context:.\\
4102      The ~ key ~ 'mini-env' ~ need ~ '\c_backslash_str miniright'.
4103    }
4104 \msg_new:nnn { enumext } { wrong-miniright-place }
4105    {
4106      Wrong ~ place ~ for ~ '\c_backslash_str miniright' ~ \msg_line_context:.~ \\
4107      Works ~ in ~ 'enumext' ~ and ~ 'keyans' ~ with ~ key ~ 'mini-env'.
4108    }
4109 \msg_new:nnn { enumext } { wrong-miniright-use }
4110    {
4111      Wrong ~ use ~ for ~ '\c_backslash_str miniright' ~ \msg_line_context:.~ \\
4112      '\c_backslash_str miniright' ~ need ~ a ~ key ~ 'mini-env'.
4113    }
```

Messages used by enumext* and keyans* environments.

```
4114 \msg_new:nnn { enumext } { nested }
4115    {
4116      The ~ starred ~ environment ~ can't ~ be ~ nested ~ \msg_line_context:.
4117    }
4118 \msg_new:nnn { enumext } { item-joined }
4119    {
4120      Items ~ joined ~ (#1) ~ > ~ #2  ~ columns ~\msg_line_context:.
4121    }
4122 \msg_new:nnn { enumext } { item-joined-columns }
4123    {
4124      Not ~ space ~ to ~ join ~ items ~ (#1) ~ > ~ #2 ~\msg_line_context:.
4125    }
```

10.42   Finish package

Finish package implementation.

```
4126 \file_input_stop:
4127 ⟨/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.