# enumext
## ENUMERATE EXERCISE SHEETS

### V1.0   2024-05-21[*]

©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 2024-05-21.
[†]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 10pt, 11pt and 12pt.

# 1 Introduction

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

4. Question with image and label below:



(A)      (B)      (C)



(D)      (E)

5. Question with image on left side:

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



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

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

* 4. (B)      *
* 5. (D), "other note"      *

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

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

## 1.1 Description and usage

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

💣 This package is NOT intend to replace the enumerate environment nor replace the powerful enumitem[5], the approach is intended to work without hindering either of them.

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

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

The package is loaded in the usual way:

```
\usepackage{enumext}
```

## 1.2 The concept of left margin

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

Usually we don't want the `list` to go beyond the left margin of the page, but since these four values are related, that causes a problem. The enumitem[5] package adds the `\labelindent` parameter to solve some of these problems. A simplified representation of this in the figure 1.

Figure 1: Representation of horizontal lengths in enumitem.

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

Figure 2: Representation of horizontal lengths concept in enumext.

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

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

## 1.3 User interface

The user interface consists in enumext, enumext*, keyans, keyans* and keyanspic environments, `\anskey`, `\item*` and `\anspic*` commands to ⟨*stored content*⟩, `\getkeyans` command to get the individual ⟨*stored content*⟩, `\printkeyans` to print all ⟨*stored content*⟩, `\miniright` for minipage and `\setenumext` to config all [⟨*key = val*⟩] options.

### 1.3.1 Internal counters

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

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

### 1.3.2 Support for multicol

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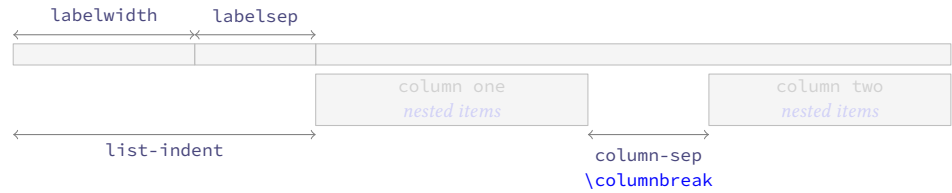


Figure 4: Representation of the two column output for a nested level in `enumext` environment.

The *"non starred"* version of the `multicols` environment is always used together with the `\raggedcolumns` command and is controlled by `columns` and `columns-sep` keys. The environment is available for all nesting levels, and can can together with the `mini-env` key. If you need to force a start a new column `\columnbreak` must be used (see §3.5).

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

### 1.3.3 Support for minipage

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



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

The `minipage` environments (left and right) is always used with *"aligned on top"* [t], the `minipage` environment on the *"right side"* always starts with `\centering`. It can be used at all nesting levels and is controlled by `mini-env` and `mini-sep` keys. In order to switch from the *"left"* side `minipage` environment to the *"right"* side one must use the command `\miniright` (see §3.6).

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

This package provides a user interface like the `enumitem`[5] package to customize the references which is activated by the `ref` key (§3.1), the standard LaTeX `\label` and `\ref` commands work as usual. It also provides an *"internal reference"* system for the *"stored content"* by means of the key `save-ref` (§4.2) when the key `save-ans` (§4.1) is active.

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

### 1.3.5 Support for `\footnote`

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

The best way to solve this is to use Jean-François Burnol `footnotehyper`[8] package, it will support keeping the links if `hyperref` is loaded with the `hyperfootnotes=true` option (default) and will show the output numbered at the bottom of the page (as opposed to how it is displayed in the `minipage` environment). The way to load it is as follows:

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

## 2 The environment `enumext`

enumext
enumext*

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

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

The `enumext` is an *"enumerated list"* environment that works in the same way as the standard `enumerate` environment provided by LaTeX, `\item` and `\item[`⟨*custom*⟩`]` commands work in the usual way.

The environment can be nested with at most *"four levels"* and the options can be configured globally using `\setenumext` command and locally using `[`⟨*key = val*⟩`]` in the environment.

**Example**

1. This text is in the first level.

   (a) This text is in the second level.

      i. This text is in the third level.

         A. This text is in the fourth level.

   X This text is in the first level.

⋆ 2. This text is in the first level.

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

## 2.1 The `\item*` in enumext

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

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

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

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

### 2.1.1 Keys for `\item*` in enumext

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

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

`item-pos*` = {⟨*rigid length | dim expression*⟩}                default: *by levels*

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

## 3 The command `\setenumext`

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

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

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

💣 It should be kept in mind that using any ⟨*key*⟩ that sets a *rubber lengths* or *rigid lengths* for vertical or horizontal space on a level will influence the vertical and horizontal space for *inners levels* and `keyans` and `keyanspic` environments. All ⟨*keys*⟩ related to vertical or horizontal spacing accept a *"skip"* or *"dim"* expression if passed between braces, i.e. you do not need to use `\dimeval` or `\dimexpr` to perform calculations.

## 3.1 Keys for `label` and `ref`

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

## 3.2 Keys for spaces

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

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

### 3.2.1 Vertical spaces

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

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

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

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

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

Set the *vertical space* added, beyond `topsep`, to the "top" and "bottom" of the entire environment if the environment instance is preceded by a *"blank line"* or `\par` command. Internally sets the value of `\partopsep` for the current level. The default values for first and second level are `2.0pt plus 1.0pt minus 1.0pt`, for third and fourth level are `1.0pt minus 1.0pt`.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

### 3.2.2 Horizontal spaces

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

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

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

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

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

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

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

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

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

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

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

### 3.3 Keys for add code

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

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

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

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

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

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

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

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

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

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

### 3.4 Keys for `start`, `series` and `resume`

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

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

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

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

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

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

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

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

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

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

### 3.5 Keys for `multicols`

columns = {⟨*integer*⟩} default: *1*

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

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

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

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

### 3.6 Keys for `minipage`

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

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

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

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

#### 3.6.1 The command `\miniright`

`\miniright`
`\miniright*`

The `\miniright` command close the `minipage` environment on the *"left side"* and opens the `minipage` environment on the *"right side"* by starting it with the `\centering` command. It must be placed *"after"* the last `\item` of the current environment and *"before"* starting the material to be placed on the *"right side"*. The *starred version* '*\**' inhibits the use of `\centering` command i.e. the usual LaTeX justification is maintained in the `minipage` on the *"right side"*.

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

#### 3.6.2 The key `miniright`

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

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

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

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

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

## 4 The storage system

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

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

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

### 4.1 Keys for storage

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

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

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

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

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

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

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

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

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

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

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

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

### 4.2 Keys for internal `label` and `ref`

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

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

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

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

### 4.3 Keys for debugging and checking

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

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

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

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

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

Enables the *checking answer* mechanism. This key works under the logic that each question will contain *"only one answer"*, it is intended to be used in conjunction with `no-store` key.

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

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

### 4.4 The command `\anskey`

`\anskey`    `\anskey{⟨content⟩}`

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

**Example**

★ 1. Text containing our instructions or questions.

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

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

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

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

### 4.5.1 The \item* in keyans

\item*
```
\item*
\item*[⟨content⟩]
```

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

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

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

**Example**

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

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

      Some text
    \end{keyans}
\end{enumext}
```

1. Text containing a question.

   (A) Choice
   * (B) Correct choice
   (C) Choice
   (D) Choice

2. Text containing a question and image.

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



Some text

## 4.6 The environment `keyanspic`

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

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



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

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

### 4.6.1 The command `\anspic`

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

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

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

**Example**

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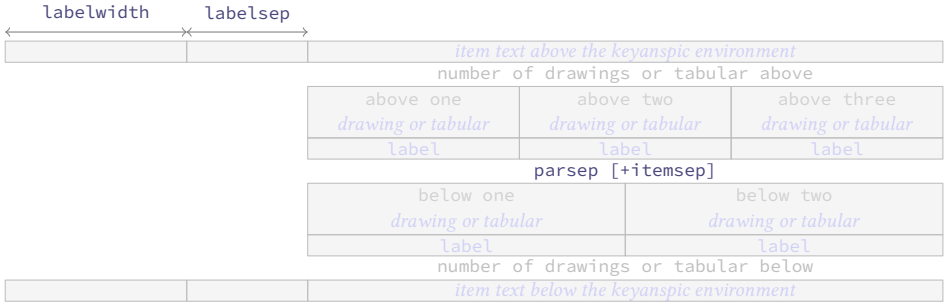
1. Question with images.



(A)



(B)



(C)



(D)



* (E)[note]
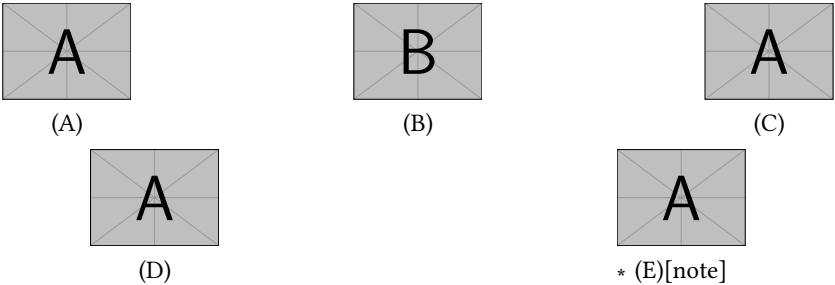
## 4.7 Printing stored content

### 4.7.1 The command \getkeyans

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

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

The *"content"* can only be accessed *"after"* it is stored, if the ⟨*store name*⟩ does not exist the command will return an error. The form taken by the argument ⟨*store name* : *position*⟩ is the same as that used to generate the internal *"label and ref"* system when save-ref key are active, so to refer to a stored *"content"*. For example \getkeyans{test:4} will return the *"stored content"* at position 4 of the environment in which the key save-ans=test was set.

### 4.7.2 The command \printkeyans

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

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

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

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

**Example**

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

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

  \item Related to Linux

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

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

\printkeyans{sample}
```

1. Factor $3x + 3y + 3z$.
   [1] $3(x+y+z)$
2. True False
   (a) LaTeX2e is cool?
       [2] Very True!
3. Related to Linux
   (a) You use linux?

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

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

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

## 5 Full examples

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

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

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

**Example 1**

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

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

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

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

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

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

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

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

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

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

**Example 2**

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

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

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

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

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

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

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

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

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

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

---

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

**Example 3**

A *"simple multiple choice"* test 🗒.

1. First type of questions
   - Ⓐ 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 :) 🗒.

 Factor $x^2 - 2x + 1$

 Factor $3x + 3y + 3z$

The following questions need to be cuaqtified :)

 True False

(a) $\alpha > \delta$

(b) LaTeX2e is cool?

 Related to Linux

(a) You use linux?

(b) Usually uses the package manager?

(c) Rate the following package and class

   i. `xsim-exam`

   ii. `xsim`

   iii. `exsheets`

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

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

**Example 5**

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

| **1** | |
|---|---|

Which choice best describes what happens in the passage?

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

| **3** | |
|---|---|

Which choice best describes what happens in the passage?

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

| **2** | |
|---|---|

Which choice best describes what happens in the passage?

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

| **4** | |
|---|---|

Which choice best describes what happens in the passage?

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

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

# 6   The way of non-enumerated lists

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

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

### Fake `itemize` environment

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

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

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

### Fake `description` environment

Here we set `label={}` and `list-indent=2.5`em,`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=0`pt 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 2024-05-21** – 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    {2024-05-21}
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.

```
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 (§10.8) and then modified by the function \__enumext_label_style:Nnn used by label key (§10.11).

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

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

\c__enumext_counter_style_tl
\l__enumext_ref_key_arg_tl
\l__enumext_ref_the_count_tl
\l__enumext_the_counter_X_tl
\l__enumext_renew_the_count_X_tl

Internal variables used by ref key (§10.11).

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

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

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

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).

```
48  \int_new:N  \g__enumext_resume_int
49  \int_new:N  \g__enumext_resume_vii_int
50  \tl_new:N   \l__enumext_resume_name_tl
51  \bool_new:N \l__enumext_resume_active_bool
52  \tl_new:N   \g__enumext_item_symbol_tl
53  \tl_new:N   \g__enumext_standar_series_tl
54  \tl_new:N   \g__enumext_starred_series_tl
```

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

\l__enumext_current_widest_dim
\g__enumext_counter_styles_tl
\g__enumext_widest_label_tl
\l__enumext_label_width_by_box

The variable \l__enumext_current_widest_dim stores the current label width, the variable \g__enumext_counter_styles_tl stores the default ⟨*label style*⟩ and the variable \g__enumext_widest_label_tl the label width. These variables are used by widest (§10.12) and label (§10.10) keys.

```
55  \dim_new:N \l__enumext_current_widest_dim
56  \tl_new:N  \g__enumext_counter_styles_tl
57  \tl_new:N  \g__enumext_widest_label_tl
58  \box_new:N \l__enumext_label_width_by_box
```

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

\l__enumext_leftmargin_tmp_X_bool
\l__enumext_leftmargin_tmp_X_dim
\l__enumext_leftmargin_X_dim
\l__enumext_itemindent_X_dim

The boolean variable \l__enumext_leftmargin_tmp_X_bool and the dimensional variable \l__enumext_leftmargin_tmp_X_dim are used by the list-indent key (§10.14).

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

```
59  \cs_set_protected:Npn \__enumext_tmp:n #1
60    {
61      \bool_new:c { l__enumext_leftmargin_tmp_#1_bool }
62      \dim_new:c  { l__enumext_leftmargin_tmp_#1_dim }
63      \dim_new:c  { l__enumext_leftmargin_#1_dim      }
64      \dim_new:c  { l__enumext_itemindent_#1_dim      }
65    }
66  \clist_map_inline:nn { i, ii, iii, iv, v, vi, vii, viii } { \__enumext_tmp:n {#1} }
```

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

\l__enumext_multicols_above_X_skip
\l__enumext_multicols_below_X_skip

Internal variables used by columns key §10.18).

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

(*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).

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

(*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

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

(*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* (§10.34.1) for the keyanspic environment.

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

```
97  \bool_new:N \l__enumext_store_active_bool
98  \tl_new:N   \l__enumext_store_name_tl
99  \tl_new:N   \g__enumext_store_name_tl
100 \tl_new:N   \l__enumext_store_anskey_arg_tl
101 \int_new:N  \l__enumext_store_columns_join_int
102 \tl_new:N   \l__enumext_store_keyans_label_tl
103 \tl_new:N   \l__enumext_store_keyans_item_opt_tl
104 \tl_new:N   \l__enumext_keyans_item_opt_tl
105 \tl_new:N   \l__enumext_keyans_tmpa_tl
106 \tl_new:N   \l__enumext_keyans_tmpb_tl
107 \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 (§10.39).

```
108 \tl_new:N  \l__enumext_setkey_tmpa_tl
109 \tl_new:N  \l__enumext_setkey_tmpb_tl
110 \int_new:N \l__enumext_setkey_tmpa_int
111 \seq_new:N \l__enumext_setkey_tmpa_seq
112 \seq_new:N \l__enumext_setkey_tmpb_seq
```

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

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

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

```
113 \cs_set_protected:Npn \__enumext_tmp:n #1
114   {
115     \tl_new:c { l__enumext_store_opt_#1_tl              }
116     \tl_new:c { l__enumext_print_keyans_#1_tl           }
117     \bool_new:c { l__enumext_store_columns_#1_bool      }
118     \int_new:c { l__enumext_store_columns_#1_int        }
119     \bool_new:c { l__enumext_store_columns_sep_#1_bool  }
120     \dim_new:c { l__enumext_store_columns_sep_#1_dim    }
121     \bool_new:c { l__enumext_store_upper_level_#1_bool  }
122   }
123 \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).

```
124 \bool_new:N \l__enumext_show_answer_bool
125 \bool_new:N \l__enumext_show_position_bool
126 \tl_new:N   \l__enumext_mark_ref_sym_tl
127 \tl_new:N   \l__enumext_mark_answer_sym_tl
128 \str_new:N  \l__enumext_mark_position_str
```

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

\l__enumext_keyans_pic_body_seq
\l__enumext_keyans_pic_width_dim
\l__enumext_keyans_pic_above_int
\l__enumext_keyans_pic_below_int
\l__enumext_keyans_pic_above_skip

Internal variables used by keyanspic environment (§10.34.2).

```
129 \seq_new:N  \l__enumext_keyans_pic_body_seq
130 \dim_new:N  \l__enumext_keyans_pic_width_dim
131 \int_new:N  \l__enumext_keyans_pic_above_int
132 \int_new:N  \l__enumext_keyans_pic_below_int
133 \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
\l__enumext_check_starred_cmd_int
\g__enumext_check_starred_cmd_int
\g__enumext_count_item_anskey_int
\g__enumext_count_item_number_int

Internal variables used by *"check answer"* mechanism (§10.23) used by the check-ans and no-store keys and check for starred commands \item* in keyans ans keyans environments and \anspic* in keyanspoc environment.

```
134 \bool_new:N \l__enumext_store_ans_bool
135 \bool_new:N \l__enumext_check_ans_bool
136 \bool_new:N \g__enumext_check_ans_show_bool
137 \bool_new:N \g__enumext_check_ans_show_h_bool
138 \tl_new:N   \g__enumext_check_ans_item_tl
139 \int_new:N  \l__enumext_check_starred_cmd_int
140 \int_new:N  \g__enumext_check_starred_cmd_int
141 \int_new:N  \g__enumext_count_item_anskey_int
142 \int_new:N  \g__enumext_count_item_number_int
143 \int_new:N  \g__enumext_standar_star_env_int
144 \int_new:N  \g__enumext_starred_star_env_int
145 \int_new:N  \g__enumext_starred_keyans_star_env_int
146 \int_new:N  \g__enumext_standar_keyans_star_env_int
147 \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.

```
148 \bool_new:N \l__enumext_hyperref_bool
149 \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.

```
150 \tl_new:N \l__enumext_newlabel_arg_one_tl
151 \tl_new:N \l__enumext_newlabel_arg_two_tl
152 \tl_new:N \l__enumext_store_write_aux_file_tl
153 \cs_set_protected:Npn \__enumext_tmp:n #1
154   {
155     \tl_new:c { l__enumext_label_copy_#1_tl }
156   }
157 \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.

```
158 \int_new:N \g__enumext_footnote_int
159 \seq_new:N \g__enumext_footnote_arg_seq
160 \seq_new:N \g__enumext_footnote_int_seq
```

(*End of definition for* \g__enumext_footnote_int, \g__enumext_footnote_arg_seq, *and* \g__enumext_footnote_int_-seq.)

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

Internal variables used by enumext* and keyans* environments.

```
161 \cs_set_protected:Npn \__enumext_tmp:n #1
162   {
163     \bool_new:c { l__enumext_item_starred_#1_bool    }
164     \int_new:c  { l__enumext_item_column_pos_#1_int  }
165     \int_new:c  { g__enumext_item_count_all_#1_int   }
166     \int_new:c  { l__enumext_joined_item_#1_int      }
167     \int_new:c  { l__enumext_joined_item_aux_#1_int  }
168     \int_new:c  { l__enumext_tmpa_#1_int             }
169     \box_new:c  { l__enumext_item_text_#1_box        }
170     \dim_new:c  { l__enumext_joined_width_#1_dim     }
171     \dim_new:c  { l__enumext_item_width_#1_dim       }
172     \tl_new:c   { g__enumext_item_symbol_aux_#1_tl   }
173     \str_new:c  { l__enumext_align_label_#1_str      }
174     \bool_new:c { g__enumext_minipage_active_#1_bool }
175     \tl_new:c   { g__enumext_miniright_code_#1_tl    }
176     \bool_new:c { g__enumext_minipage_center_#1_bool }
177     \dim_new:c  { g__enumext_minipage_right_#1_dim   }
178     \skip_new:c { g__enumext_minipage_right_#1_skip  }
179   }
180 \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.

```
181 \clist_const:Nn \c__enumext_all_envs_clist
182   {
183     {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv},
184     {keyans}{v}, {enumext*}{vii}, {keyans*}{viii}
185   }
```

(*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.

```
186 \cs_new_protected:Npn \__enumext_at_begin_document:n #1
187   {
188     \hook_gput_code:nnn {begindocument} {enumext} { #1 }
189   }
```

(*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.

```
190 \cs_new_protected:Npn \__enumext_after_env:nn #1 #2
191   {
192     \hook_gput_code:nnn {env/#1/after} {enumext} {#2}
193   }
```

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

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

```
194 \cs_new:Nn \__enumext_level:
195   {
196     \int_to_roman:n { \l__enumext_level_int }
197   }
```

(*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?.

```
198 \prg_new_protected_conditional:Npnn \__enumext_if_is_int:n #1 { T, F, TF }
199   {
200     \regex_match:nnTF { ^[\+\-]?[\d]+$ } {#1} % $
201       { \prg_return_true: }
202       { \prg_return_false: }
203   }
```

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

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

```
204 \cs_new_protected:Nn \__enumext_regex_counter_style:
205   {
206     \tl_map_inline:Nn \c__enumext_counter_style_tl
207       {
208         \regex_replace_once:nnN { \c{##1}\* }
209           { \c{##1}\cB{\u{l__enumext_ref_the_count_tl}\cE} } \l__enumext_ref_key_arg_tl
210       }
211   }
```

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

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

```
212 \cs_new:Npn \__enumext_show_length:nnn #1 #2 #3
213   {
214     * ~ #2
215     \prg_replicate:nn { 14 - \str_count:n {#2} } { ~ }
216       = ~ \use:c { #1_use:c } { l__enumext_#2_#3_#1 } \\
217   }
```

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

`\__enumext_zero_count_level:` Internal function used by `check-ans` key.

```
218 \cs_set_protected:Nn \__enumext_zero_count_level:
219   {
220     \cs_set_protected:Npn \__enumext_tmp:n ##1
221       {
222         \int_gzero:c { g__enumext_count_level_##1_int }
223       }
224     \clist_map_inline:nn { i, ii, iii, iv, vii } { \__enumext_tmp:n {##1} }
225   }
```

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

`\__enumext_current_env_set_bool:`
`\__enumext_check_first_level:`
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.

```
226 \cs_new_protected:Nn \__enumext_current_env_set_bool:
227   {
228     \str_case:en { \@currenvir }
229       {
230         {enumext}
231           {
232             \bool_lazy_and:nnT
233               { \bool_not_p:n { \g__enumext_standar_bool } }
234               { \int_compare_p:nNn { \l__enumext_level_h_int } = { \c_zero_int } }
235               {
236                 \bool_gset_true:N \g__enumext_standar_bool
237                 \int_gset:Nn \g__enumext_standar_star_env_int { \inputlineno }
238                 \typeout{working-on-enumext}
239               }
240           }
241         {enumext*}
242           {
243             \bool_lazy_and:nnT
244               { \bool_not_p:n { \g__enumext_starred_bool } }
245               { \int_compare_p:nNn { \l__enumext_level_int } = { \c_zero_int } }
246               {
247                 \bool_gset_true:N \g__enumext_starred_bool
248                 \int_gset:Nn \g__enumext_starred_star_env_int { \inputlineno }
249                 \typeout{working-on-enumext*}
250               }
251           }
252       }
253   }
```

The function `\__enumext_check_first_level:` will set the variables `\l__enumext_standar_-level_one_bool` and `\l__enumext_standar_level_one_bool` to "*true*" only if the environment is not nested and we are at the "*first level*" of it.

```
254 \cs_new_protected:Nn \__enumext_check_first_level:
255   {
256     \bool_lazy_all:nT
257       {
258         { \bool_if_p:N \g__enumext_standar_bool }
259         { \int_compare_p:nNn { \l__enumext_level_int } = { 1 } }
260         { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
261       }
262       {
263         \bool_set_true:N \l__enumext_standar_level_one_bool
264       }
265     \bool_lazy_all:nT
266       {
267         { \bool_if_p:N \g__enumext_starred_bool }
268         { \int_compare_p:nNn { \l__enumext_level_h_int } = { 1 } }
269         { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
270       }
271       {
272         \bool_set_true:N \l__enumext_starred_level_one_bool
273       }
274   }
```

(*End of definition for* `\__enumext_current_env_set_bool:` *and* `\__enumext_check_first_level:`.)

This functions is passed to the `\__enumext_safe_exec:` function in the definition of the enumext environment (pag 77) and to the `\__enumext_safe_exec_vii:` function in the definition of the enumext* environment (pag 89).

## 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.

```
275 \__enumext_at_begin_document:n
276   {
277     \cs_new_eq:NN \__enumext_start_list:nn \list
278     \cs_new_eq:NN \__enumext_stop_list: \endlist
279     \cs_new_eq:NN \__enumext_item_std:w \item
280   }
```

*(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.

```
281 \__enumext_at_begin_document:n
282   {
283     \cs_new_eq:NN \__enumext_minipage:w \minipage
284     \cs_new_eq:NN \__enumext_endminipage: \endminipage
285   }
```

*(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.

```
286 \hook_gput_code:nnn { begindocument } { enumext } { \__enumext_after_hyperref: }
287 \hook_gset_rule:nnnn { begindocument } { enumext } { after } { hyperref }
```

`\__enumext_after_hyperref:`
`\__enumext_hypertarget:nn`
`\__enumext_phantomsection:`

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

```
288 \cs_new_protected:Nn \__enumext_after_hyperref:
289   {
290     \IfPackageLoadedTF { hyperref }
291       {
292         \msg_info:nnn { enumext } { package-load } { hyperref }
293         \bool_set_true:N \l__enumext_hyperref_bool
294         \IfHyperBoolean{hyperfootnotes}
295           {
296             \typeout{hyperfootnotes=true}
297             \bool_set_true:N \l__enumext_footnotes_key_bool
298           }
299           { \typeout{hyperfootnotes=false} }
300       }
301       {  }
```

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.

```
302    \bool_if:NT \l__enumext_footnotes_key_bool
303      {
304        \IfPackageLoadedTF { footnotehyper }
305          {
306            \msg_info:nnn { enumext } { package-load } { footnotehyper }
307          }
308          {
309            \typeout{No ~ footnotehyper ~ load}
310            \typeout{Load ~ and  ~ use  ~ \string\makesavenoteenv{enumext*}}
311            \bool_set_false:N \l__enumext_footnotes_key_bool
312          }
313      }
```

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.

```
314    \bool_if:NTF \l__enumext_hyperref_bool
315      {
316        \cs_new_eq:NN \__enumext_hypertarget:nn \hypertarget
317        \cs_new_eq:NN \__enumext_phantomsection: \phantomsection
318      }
319      {
320        \cs_new_eq:NN \__enumext_hypertarget:nn \use_none:nn
321        \cs_new_eq:NN \__enumext_phantomsection: \prg_do_nothing:
322      }
323    }
```

(*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.

```
324 \cs_new_protected:Npn \__enumext_newlabel:nn #1 #2
325   {
326     \protected@write \@auxout { }
327       {
328         \token_to_str:N \newlabel {#1}
329           {
330             {#2}
331             \bool_if:NT \l__enumext_hyperref_bool
332               { { \thepage } {#2} {#1} }
333             { }
334           }
335       }
336     \__enumext_hypertarget:nn {#1} { }
337     \__enumext_phantomsection:
338   }
```

(*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.

```
339 \cs_new_protected:Npn \__enumext_define_counters:Nn #1 #2
340   {
341     \cs_if_exist:cTF { c@ #2 }
342       { \msg_fatal:nnn { enumext } { counters }{ #2 } }
343       {
344         \tl_set:Nn #1 { #2 }
```

```
345                \newcounter { #2 }
346            }
347      }
```

(*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.

```
348  \__enumext_define_counters:Nn \l__enumext_counter_i_tl     { enumXi     }
349  \__enumext_define_counters:Nn \l__enumext_counter_ii_tl    { enumXii    }
350  \__enumext_define_counters:Nn \l__enumext_counter_iii_tl   { enumXiii   }
351  \__enumext_define_counters:Nn \l__enumext_counter_iv_tl    { enumXiv    }
352  \__enumext_define_counters:Nn \l__enumext_counter_v_tl     { enumXv     }
353  \__enumext_define_counters:Nn \l__enumext_counter_vi_tl    { enumXvi    }
354  \__enumext_define_counters:Nn \l__enumext_counter_vii_tl   { enumXvii   }
355  \__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.

```
356  \cs_new_protected:Npn \__enumext_register_counter_style:Nn #1 #2
357    {
358      \tl_const:cn { c__enumext_widest_ \cs_to_str:N #1 _tl } {#2}
359      \tl_gput_right:Nn \g__enumext_counter_styles_tl {#1}
360    }
361  \__enumext_register_counter_style:Nn \arabic { 0 }
362  \__enumext_register_counter_style:Nn \Alph   { M }
363  \__enumext_register_counter_style:Nn \alph   { m }
364  \__enumext_register_counter_style:Nn \Roman  { VIII }
365  \__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.

```
366  \cs_new_protected:Npn \__enumext_label_width_by_box:Nn #1 #2
367    {
368      \hbox_set:Nn \l__enumext_label_width_by_box {#2}
369      \dim_set:Nn #1 { \box_wd:N \l__enumext_label_width_by_box }
370    }
371  \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.

```
372  \cs_new_protected:Npn \__enumext_label_style:Nnn #1 #2 #3
373    {
374      \tl_clear_new:N #1
375      \tl_put_right:Ne #1 { \tl_trim_spaces:n {#3} }
376      \tl_gset_eq:NN \g__enumext_widest_label_tl #1
377      \tl_map_inline:Nn \g__enumext_counter_styles_tl
378        {
379          \tl_replace_all:Nne #1 { ##1* } { \exp_not:N ##1 {#2} }
380          \tl_greplace_all:Nne \g__enumext_widest_label_tl { ##1* }
381            { \tl_use:c { c__enumext_widest_ \cs_to_str:N ##1 _tl } }
382        }
383      \__enumext_label_width_by_box:Nn \l__enumext_current_widest_dim
384        { \tl_use:N \g__enumext_widest_label_tl }
385      \tl_set_eq:cN { the #2 } #1
386    }
387  \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.

```
388  \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
389    {
390      \keys_define:nn { enumext / #1 }
391        {
392          font        .tl_set:c   = { l__enumext_label_font_style_#2_tl },
393          font        .value_required:n = true,
394          labelsep    .dim_set:c   = { l__enumext_labelsep_#2_dim },
395          labelsep    .initial:n  = {0.3333em},
396          labelsep    .value_required:n = true,
397          labelwidth  .dim_set:c   = { l__enumext_labelwidth_#2_dim },
398          labelwidth  .value_required:n = true,
399          wrap-label  .cs_set_protected:cp = { __enumext_wrapper_label_#2:n } ##1,
400          wrap-label  .initial:n  = {##1},
401          wrap-label  .value_required:n = true,
402          wrap-label* .code:n = {
403                              \bool_set_true:c { l__enumext_wrap_label_opt_#2_bool }
404                              \keys_set:nn { enumext / #1 } { wrap-label = {##1} }
405                            },
406          wrap-label* .value_required:n = true,
407        }
408    }
409  \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.

```
410  \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
411    {
412      \keys_define:nn { enumext / #1 }
413        {
414          align .choice:,
415          align / left   .code:n =
416                          {
417                            \tl_clear:c { l__enumext_label_fill_left_#2_tl  }
418                            \tl_set:cn  { l__enumext_label_fill_right_#2_tl } { \hfill }
419                          },
420          align / right  .code:n =
421                          {
422                            \tl_set:cn  { l__enumext_label_fill_left_#2_tl  } { \hfill }
423                            \tl_clear:c { l__enumext_label_fill_right_#2_tl }
424                          },
425          align / center .code:n =
426                          {
427                            \tl_set:cn { l__enumext_label_fill_left_#2_tl  } { \hfill }
428                            \tl_set:cn { l__enumext_label_fill_right_#2_tl } { \hfill }
429                          },
430          align .initial:n  = left,
431          align .value_required:n  = true,
432        }
433    }
434  \clist_map_inline:nn
435    {
436      {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {keyans}{v}
437    }
438    { \__enumext_tmp:nn #1 }

439  \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
440    {
441      \keys_define:nn { enumext / #1 }
442        {
443          align .choice:,
444          align / left   .code:n = \str_set:cn { l__enumext_align_label_#2_str } { l },
```

```
445        align / right  .code:n = \str_set:cn { l__enumext_align_label_#2_str } { r },
446        align / center .code:n = \str_set:cn { l__enumext_align_label_#2_str } { c },
447        align .initial:n = left,
448        align .value_required:n = true,
449      }
450    }
451  \clist_map_inline:nn { {enumext*}{vii}, {keyans*}{viii} } { \__enumext_tmp:nn #1 }
```

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

## 10.11 Setting `label` and `ref` keys

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

### 10.11.1 Define and set `label` and `ref` keys for enumext environment

<div style="float:left">

label
ref
`\l__enumext_label_i_tl`
`\l__enumext_label_ii_tl`
`\l__enumext_label_iii_tl`
`\l__enumext_label_iv_tl`

</div>

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

```
452  \cs_set_protected:Npn \__enumext_tmp:nnn #1 #2 #3
453    {
454      \keys_define:nn { enumext / #1 }
455        {
456          label .code:n    = {
457                               \__enumext_label_style:cvn { l__enumext_label_#2_tl }
458                               { l__enumext_counter_#2_tl } {##1}
459                               \dim_set_eq:cN  { l__enumext_labelwidth_#2_dim }
460                               \l__enumext_current_widest_dim
461                             },
462          label .initial:n = #3,
463          label .value_required:n = true,
464          ref   .code:n    = \__enumext_standar_ref:n {##1},
465          ref   .value_required:n = true,
466        }
467    }
468  \__enumext_tmp:nnn { level-1 } {   i } { \arabic*.}
469  \__enumext_tmp:nnn { level-2 } {  ii } { (\alph*) }
470  \__enumext_tmp:nnn { level-3 } { iii } { \roman*. }
471  \__enumext_tmp:nnn { level-4 } {  iv } { \Alph*.  }
```

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

<div style="float:left">

`\__enumext_standar_ref:n`
`\__enumext_standar_ref:`

</div>

The `\__enumext_standar_ref:n` first we will pass the key argument to `\l__enumext_ref_key_-arg_tl` and we will analyze its state, if it is not *empty* we will make a copy of the current counter in `\l__enumext _ref_the_count_tl` and we will execute the function `\__enumext_regex_counter_-style:` which will return the modified `\l__enumext_ref_key_arg_tl` and we make the value of `\l__enumext_ref_the_count_tl` the same as that `\l__enumext_the_counter_X_tl` which contains `\theenumX` and finally we set `\l__enumext_renew_the_count_X_tl` with the renewed command.

```
472  \cs_new_protected:Npn \__enumext_standar_ref:n #1
473    {
474      \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
475      \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
476        {
477          \msg_error:nnn { enumext } { key-ref-empty } { enumext }
478        }
479        {
480          \tl_set_eq:Nc
481            \l__enumext_ref_the_count_tl { l__enumext_counter_ \__enumext_level: _tl }
482          \__enumext_regex_counter_style:
483          \tl_set_eq:Nc
484            \l__enumext_ref_the_count_tl { l__enumext_the_counter_ \__enumext_level: _tl }
485          \tl_put_right:ce { l__enumext_renew_the_count_ \__enumext_level: _tl }
486            {
487              \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl }
488                { \exp_not:V \l__enumext_ref_key_arg_tl }
489            }
490        }
491    }
```

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

```
492  \cs_new_protected:Nn \__enumext_standar_ref:
493    {
494      \tl_if_empty:cF { l__enumext_renew_the_count_ \__enumext_level: _tl }
495        {
496          \tl_use:c { l__enumext_renew_the_count_ \__enumext_level: _tl }
497        }
498    }
```

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

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

label
ref
\l__enumext_label_vii_tl
\l__enumext_label_viii_tl

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

```
499  \cs_set_protected:Npn \__enumext_tmp:nnn #1 #2 #3
500    {
501      \keys_define:nn { enumext / #1 }
502        {
503          label .code:n    = {
504                             \__enumext_label_style:cvn { l__enumext_label_#2_tl }
505                               { l__enumext_counter_#2_tl } {##1}
506                             \dim_set_eq:cN  { l__enumext_labelwidth_#2_dim }
507                               \l__enumext_current_widest_dim
508                           },
509          label .initial:n = #3,
510          label .value_required:n = true,
511          ref    .code:n    = \__enumext_starred_ref:n {##1},
512          ref    .value_required:n = true,
513        }
514    }
515  \__enumext_tmp:nnn { enumext* } {  vii } { \arabic*.}
516  \__enumext_tmp:nnn { keyans*  } { viii } { (\Alph*) }
```

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

\__enumext_starred_ref:n
\__enumext_starred_ref:

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

```
517  \cs_new_protected:Npn \__enumext_starred_ref:n #1
518    {
519      \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
520      \int_compare:nNnT { \l__enumext_level_h_int } = { 1 }
521        {
522          \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
523            {
524              \msg_error:nnn { enumext } { key-ref-empty } { enumext* }
525            }
526            {
527              \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_counter_vii_tl
528              \__enumext_regex_counter_style:
529              \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_the_counter_vii_tl
530              \tl_put_right:Ne \l__enumext_renew_the_count_vii_tl
531                {
532                  \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl }
533                    { \exp_not:V \l__enumext_ref_key_arg_tl }
534                }
535            }
536        }
537      \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
538        {
539          \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
540            {
541              \msg_error:nnn { enumext } { key-ref-empty } { keyans* }
542            }
543            {
544              \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_counter_viii_tl
545              \__enumext_regex_counter_style:
546              \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_the_counter_viii_tl
547              \tl_put_right:Ne \l__enumext_renew_the_count_vii_tl
548                {
549                  \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl }
```

```
550                        { \exp_not:V \l__enumext_ref_key_arg_tl }
551                      }
552                    }
553                  }
554              }
```

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

```
555  \cs_new_protected:Nn \__enumext_starred_ref:
556    {
557      \int_compare:nNnT { \l__enumext_level_h_int } = { 1 }
558        {
559          \tl_if_empty:NF \l__enumext_renew_the_count_vii_tl
560            {
561              \tl_use:N \l__enumext_renew_the_count_vii_tl
562            }
563        }
564      \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
565        {
566          \tl_if_empty:NF \l__enumext_renew_the_count_viii_tl
567            {
568              \tl_use:N \l__enumext_renew_the_count_viii_tl
569            }
570        }
571    }
```

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

### 10.11.3  Define and set `label` and `ref` keys for `keyans` and `keyanspic` environments

label
ref
`\l__enumext_label_v_tl`
`\l__enumext_label_vi_tl`

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

```
572  \keys_define:nn { enumext / keyans }
573    {
574      label .code:n    = {
575                          \__enumext_label_style:cvn { l__enumext_label_v_tl }
576                            { l__enumext_counter_v_tl } {#1}
577                          \dim_set_eq:cN  { l__enumext_labelwidth_v_dim }
578                            \l__enumext_current_widest_dim
579                          \__enumext_label_style:cvn { l__enumext_label_vi_tl }
580                            { l__enumext_counter_vi_tl } {#1}
581                          \dim_set_eq:cN  { l__enumext_labelwidth_v_dim }
582                            \l__enumext_current_widest_dim
583                        },
584      label .initial:n = (\Alph*),
585      label .value_required:n = true,
586      ref   .code:n    = \__enumext_keyans_ref:n {#1},
587      ref   .value_required:n = true,
588    }
```

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

`\__enumext_keyans_ref:n`
`\__enumext_keyans_ref:`

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

```
589  \cs_new_protected:Npn \__enumext_keyans_ref:n #1
590    {
591      \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
592      \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
593        {
594          \msg_error:nnn { enumext } { key-ref-empty } { keyans }
595        }
596        {
597          \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_counter_v_tl
598          \__enumext_regex_counter_style:
599          \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_the_counter_v_tl
600          \tl_put_right:Ne \l__enumext_renew_the_count_v_tl
601            {
602              \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl }
603                { \exp_not:V \l__enumext_ref_key_arg_tl }
604            }
605        }
606    }
```

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

```
607 \cs_new_protected:Nn \__enumext_keyans_ref:
608   {
609     \tl_if_empty:NF \l__enumext_renew_the_count_v_tl
610       {
611         \tl_use:N \l__enumext_renew_the_count_v_tl
612       }
613   }
```

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

## 10.12 Setting start and widest keys

\__enumext_start_from:NNn
\__enumext_start_from:ccn

The function \__enumext_start_from:NNn used by the start key take three arguments:

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

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

```
614 \cs_new_protected:Npn \__enumext_start_from:NNn #1 #2 #3
615   {
616     \__enumext_if_is_int:nTF { #3 }
617       {
618         \int_set:Nn #2 {#3}
619       }
620       {
621         \regex_match:nVT { \c{Alph} | \c{alph} } {#1}
622           { \int_set:Nn #2 { \int_from_alph:n {#3} } }
623         \regex_match:nVT { \c{Roman} | \c{roman} } {#1}
624           { \int_set:Nn #2  { \int_from_roman:n {#3} } }
625       }
626   }
627 \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.

```
628 \cs_new_protected:Npn \__enumext_widest_from:nNNn #1 #2 #3 #4
629   {
630     \__enumext_if_is_int:nTF {#4}
631       {
632         \setcounter{enumX#1} { #4 }
633       }
634       {
635         \regex_match:nVT { \c{Alph} | \c{alph} } {#2}
636           { \setcounter{enumX#1} { \int_from_alph:n {#4} } }
637         \regex_match:nVT { \c{Roman} | \c{roman} } {#2}
638           { \setcounter{enumX#1} { \int_from_roman:n {#4} } }
639       }
640     \__enumext_label_width_by_box:cv
641       { l__enumext_labelwidth_#1_dim } { l__enumext_label_#1_tl }
642   }
643 \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.

```
644 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
645   {
646     \keys_define:nn { enumext / #1 }
```

```
647        {
648          start  .code:n    = {
649                                \__enumext_start_from:ccn
650                                  { l__enumext_label_#2_tl }
651                                  { l__enumext_start_#2_int } {##1}
652                               },
653          start  .initial:n = 1,
654          widest .code:n    = {
655                                \__enumext_widest_from:nccn {#2}
656                                  { l__enumext_label_#2_tl }
657                                  { l__enumext_labelwidth_#2_dim } {##1}
658                               },
659          widest .value_required:n = true,
660          start  .value_required:n = true,
661        }
662    }
663  \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

<span style="float:left">topsep<br>partopsep<br>parsep<br>noitemsep<br>nosep</span>

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

```
664  \cs_set_protected:Npn \__enumext_tmp:nnnnnn #1 #2 #3 #4 #5 #6
665    {
666      \keys_define:nn { enumext / #1 }
667        {
668          topsep    .skip_set:c = { l__enumext_topsep_#2_skip },
669          topsep    .initial:n  = {#3},
670          topsep    .value_required:n = true,
671          partopsep .skip_set:c = { l__enumext_partopsep_#2_skip },
672          partopsep .initial:n  = {#4},
673          partopsep .value_required:n = true,
674          parsep    .skip_set:c = { l__enumext_parsep_#2_skip },
675          parsep    .initial:n  = {#5},
676          parsep    .value_required:n = true,
677          itemsep   .skip_set:c = { l__enumext_itemsep_#2_skip },
678          itemsep   .initial:n  = {#6},
679          itemsep   .value_required:n = true,
680          noitemsep .meta:n     = { itemsep = 0pt, parsep = 0pt },
681          noitemsep .value_forbidden:n = true,
682          nosep     .meta:n     = {
683                                    itemsep = 0pt, parsep= 0pt,
684                                    topsep = 0pt, partopsep = 0pt,
685                                  },
686          nosep     .value_forbidden:n = true,
687        }
688    }
```

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

```
689  \__enumext_tmp:nnnnnn { level-1 } { i } { 8.0pt plus 2.0pt minus 4.0pt }
690    { 2.0pt plus 1.0pt minus 1.0pt } { 4.0pt plus 2.0pt minus 1.0pt }
691    { 4.0pt plus 2.0pt minus 1.0pt }
692  \__enumext_tmp:nnnnnn { level-2 } { ii } { 4.0pt plus 2.0pt minus 1.0pt }
693    { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
694    { 2.0pt plus 1.0pt minus 1.0pt }
695  \__enumext_tmp:nnnnnn { level-3 } { iii } { 2.0pt plus 1.0pt minus 1.0pt }
696    { 1.0pt minus 1.0pt }{ 0pt }{ 2.0pt plus 1.0pt minus 1.0pt }
697  \__enumext_tmp:nnnnnn { level-4 } { iv } { 2.0pt plus 1.0pt minus 1.0pt }
698    { 1.0pt minus 1.0pt }{ 0pt }{ 2.0pt plus 1.0pt minus 1.0pt }
699  \__enumext_tmp:nnnnnn { keyans  } { v }{ 4.0pt plus 2.0pt minus 1.0pt }
700    { 2.0pt plus 1.0pt minus 1.0pt }{ 2.0pt plus 1.0pt minus 1.0pt }
701    { 2.0pt plus 1.0pt minus 1.0pt }
702  \__enumext_tmp:nnnnnn { enumext* } { vii } { 8.0pt plus 2.0pt minus 4.0pt }
703    { 2.0pt plus 1.0pt minus 1.0pt } { 4.0pt plus 2.0pt minus 1.0pt }
704    { 4.0pt plus 2.0pt minus 1.0pt }
705  \__enumext_tmp:nnnnnn { keyans* } { viii } { 4.0pt plus 2.0pt minus 1.0pt }
706    { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
707    { 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.

```
708 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
709   {
710     \keys_define:nn { enumext / #1 }
711       {
712         itemindent    .dim_set:c = { l__enumext_fake_item_indent_#2_dim },
713         itemindent    .value_required:n = true,
714         rightmargin   .dim_set:c = { l__enumext_rightmargin_#2_dim },
715         rightmargin   .value_required:n = true,
716         listparindent .dim_set:c = { l__enumext_listparindent_#2_dim },
717         listparindent .value_required:n = true,
718         list-offset   .dim_set:c = { l__enumext_listoffset_#2_dim },
719         list-offset   .value_required:n = true,
720         list-indent   .code:n    =
721                          \bool_set_true:c { l__enumext_leftmargin_tmp_#2_bool }
722                          \dim_set:cn { l__enumext_leftmargin_tmp_#2_dim } {##1},
723         list-indent   .value_required:n = true,
724       }
725   }
726 \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.

```
727 \cs_set_protected:Npn \__enumext_tmp:n #1
728   {
729     \keys_define:nn { enumext / #1 } { list-indent .initial:n  = 0pt, }
730   }
731 \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.

```
732 \cs_set_protected:Nn \__enumext_fake_item:
733   {
734     \dim_compare:nNnT
735       { \dim_use:c { l__enumext_fake_item_indent_ \__enumext_level: _dim } }
736       >
737       { \c_zero_dim }
738       {
739         \tl_set:ce { l__enumext_fake_item_indent_ \__enumext_level: _tl }
740           {
741             \exp_not:N \mode_leave_vertical:
742             \exp_not:n { \skip_horizontal:n }
743               { \dim_use:c { l__enumext_fake_item_indent_ \__enumext_level: _dim } }
744             \ignorespaces
745           }
746       }
747   }
748 \cs_set_protected:Nn \__enumext_keyans_fake_item:
749   {
750     \dim_compare:nNnT
751       { \l__enumext_fake_item_indent_v_dim } > { \c_zero_dim }
752       {
753         \tl_set:Ne \l__enumext_fake_item_indent_v_tl
754           {
755             \exp_not:N \mode_leave_vertical:
756             \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_v_dim
757           }
758       }
759   }
760 \cs_set_protected:Nn \__enumext_fake_item_vii:
761   {
762     \dim_compare:nNnT
763       { \l__enumext_fake_item_indent_vii_dim } > { \c_zero_dim }
```

```
764        {
765            \tl_set:Ne \l__enumext_fake_item_indent_vii_tl
766               {
767                  \exp_not:N \mode_leave_vertical:
768                  \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_vii_dim
769               }
770            }
771        }
772    \cs_set_protected:Nn \__enumext_fake_item_viii:
773        {
774           \dim_compare:nNnT
775               { \l__enumext_fake_item_indent_viii_dim } > { \c_zero_dim }
776               {
777                  \tl_set:Ne \l__enumext_fake_item_indent_viii_tl
778                     {
779                        \exp_not:N \mode_leave_vertical:
780                        \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_viii_dim
781                     }
782               }
783        }
```

(*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.

```
784    \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
785        {
786           \keys_define:nn { enumext / #1 }
787               {
788                  show-length .bool_set:c = { l__enumext_show_length_#2_bool },
789                  show-length .initial:n  = false,
790               }
791        }
792    \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }
```

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

## 10.16  Setting `before`, `after` and `first` keys

before
before*
after
first

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

```
793    \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
794        {
795           \keys_define:nn { enumext / #1 }
796               {
797                  before  .tl_set:c   = { l__enumext_before_no_starred_key_#2_tl },
798                  before  .value_required:n = true,
799                  before* .tl_set:c   = { l__enumext_before_starred_key_#2_tl },
800                  before* .value_required:n = true,
801                  after   .tl_set:c   = { l__enumext_after_stop_list_#2_tl },
802                  after   .value_required:n = true,
803                  first   .tl_set:c   = { l__enumext_after_list_args_#2_tl },
804                  first   .value_required:n = true,
805               }
806        }
807    \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }
```

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

### 10.16.1  Functions for `before`, `after` and `first` keys in `enumext`

\__enumext_before_args_exec:
\__enumext_before_keys_exec:
\__enumext_after_stop_list:
\__enumext_after_args_exec:

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

```
808    \cs_new_protected:Nn \__enumext_before_args_exec:
809        {
810           \tl_use:c { l__enumext_before_starred_key_ \__enumext_level: _tl }
811        }
```

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*⟩.

```
812  \cs_new_protected:Nn \__enumext_before_keys_exec:
813    {
814      \tl_use:c { l__enumext_before_no_starred_key_ \__enumext_level: _tl }
815    }
```

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

```
816  \cs_new_protected:Nn \__enumext_after_stop_list:
817    {
818      \tl_use:c { l__enumext_after_stop_list_ \__enumext_level: _tl }
819    }
```

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.

```
820  \cs_new_protected:Nn \__enumext_after_args_exec:
821    {
822      \tl_use:c { l__enumext_after_list_args_ \__enumext_level: _tl }
823    }
```

(*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.

```
824  \cs_new_protected:Nn \__enumext_before_args_exec_v:
825    {
826      \tl_use:N \l__enumext_before_starred_key_v_tl
827    }
```

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*⟩.

```
828  \cs_new_protected:Nn \__enumext_before_keys_exec_v:
829    {
830      \tl_use:N \l__enumext_before_no_starred_key_v_tl
831    }
```

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

```
832  \cs_new_protected:Nn \__enumext_after_stop_list_v:
833    {
834      \tl_use:N \l__enumext_after_stop_list_v_tl
835    }
```

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.

```
836  \cs_new_protected:Nn \__enumext_after_args_exec_v:
837    {
838      \tl_use:N \l__enumext_after_list_args_v_tl
839    }
```

(*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.

```
840  \cs_new_protected:Nn \__enumext_before_args_exec_vii:
841    {
842      \tl_use:N \l__enumext_before_starred_key_vii_tl
843    }
844  \cs_new_protected:Nn \__enumext_before_args_exec_viii:
845    {
846      \tl_use:N \l__enumext_before_starred_key_viii_tl
847    }
```

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*⟩.

```
848 \cs_new_protected:Nn \__enumext_before_keys_exec_vii:
849   {
850     \tl_use:N \l__enumext_before_no_starred_key_vii_tl
851   }
852 \cs_new_protected:Nn \__enumext_before_keys_exec_viii:
853   {
854     \tl_use:N \l__enumext_before_no_starred_key_viii_tl
855   }
```

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

```
856 \cs_new_protected:Nn \__enumext_after_stop_list_vii:
857   {
858     \tl_use:N \l__enumext_after_stop_list_vii_tl
859   }
860 \cs_new_protected:Nn \__enumext_after_stop_list_viii:
861   {
862     \tl_use:N \l__enumext_after_stop_list_viii_tl
863   }
```

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`.

```
864 \cs_new_protected:Nn \__enumext_after_args_exec_vii:
865   {
866     \tl_use:N \l__enumext_after_list_args_vii_tl
867   }
868 \cs_new_protected:Nn \__enumext_after_args_exec_viii:
869   {
870     \tl_use:N \l__enumext_after_list_args_viii_tl
871   }
```

(*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.

```
872 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
873   {
874     \keys_define:nn { enumext / #1 }
875       {
876         mini-env    .dim_set:c  = { l__enumext_minipage_right_#2_dim },
877         mini-env    .value_required:n = true,
878         mini-sep    .dim_set:c  = { l__enumext_minipage_hsep_#2_dim },
879         mini-sep    .initial:n  = 0.3333em,
880         mini-sep    .value_required:n = true,
881         columns-sep .dim_set:c  = { l__enumext_columns_sep_#2_dim },
882         columns-sep .value_required:n = true,
883         columns     .int_set:c  = { l__enumext_columns_#2_int },
884         columns     .initial:n  = 1,
885         columns     .value_required:n = true,
886       }
887   }
888 \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`.

```
889 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
890   {
891     \keys_define:nn { enumext / #1 }
892       {
893         columns     .initial:n = 2,
894         miniright   .tl_gset:c = { g__enumext_miniright_code_#2_tl },
895         miniright   .value_required:n = true,
896         miniright* .code:n    = {
```

```
897                                             \bool_gset_true:c { g__enumext_minipage_center_#2_bool }
898                                             \keys_set:nn { enumext / #1 } { miniright = {##1} }
899                                          },
900                   miniright* .value_required:n = true,
901                }
902           }
903   \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.
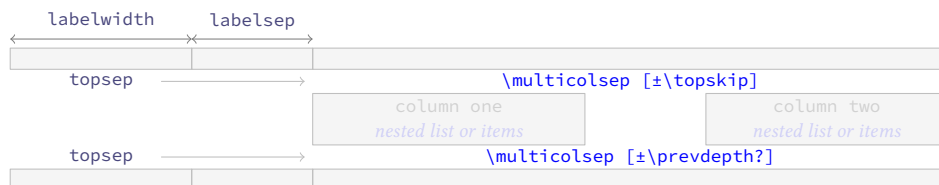


Figure 7: Representation of the vertical space in multicols for a nested level.

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*.

```
904   \cs_new_protected:Nn \__enumext_multi_set_vskip:
905     {
906       \skip_set:cn { l__enumext_multicols_above_ \__enumext_level: _skip }
907         {
908           \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
909         }
910       \skip_set:cn { l__enumext_multicols_below_ \__enumext_level: _skip }
911         {
912           \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
913         }
914       \__enumext_add_pre_parsep:
915     }
```

*(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*.

```
916   \cs_new_protected:Nn \__enumext_add_pre_parsep:
917     {
918       \int_case:nn { \l__enumext_level_int }
919         {
920           { 2 }{
921                 \skip_if_eq:nnF { \l__enumext_parsep_i_skip } { \c_zero_skip }
922                   {
923                     \skip_add:Nn \l__enumext_multicols_above_ii_skip { \l__enumext_parsep_i_skip }
924                   }
925               }
926           { 3 }{
927                 \skip_if_eq:nnF { \l__enumext_parsep_ii_skip } { \c_zero_skip }
928                   {
```

```
929                        \skip_add:Nn \l__enumext_multicols_above_iii_skip { \l__enumext_parsep_ii_skip
930                    }
931                }
932            { 4 }{
933                    \skip_if_eq:nnF { \l__enumext_parsep_iii_skip } { \c_zero_skip }
934                    {
935                    \skip_add:Nn \l__enumext_multicols_above_iv_skip { \l__enumext_parsep_iii_skip
936                    }
937                }
938            }
939        }
```

(*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*⟩.

```
940 \cs_new_protected:Nn \__enumext_multi_addvspace:
941   {
942     \__enumext_multi_set_vskip:
943     \mode_if_vertical:T
944       {
945         \skip_add:cn { l__enumext_multicols_above_ \__enumext_level: _skip }
946           {
947             \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
948           }
949         \skip_add:cn { l__enumext_multicols_below_ \__enumext_level: _skip }
950           {
951             \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
952           }
953       }
954     \par\nopagebreak
955     \addvspace{ \skip_use:c { l__enumext_multicols_above_ \__enumext_level: _skip } }
956   }
```

(*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 imple-
mentation of this function is the same as the one used in enumext.

```
957 \cs_new_protected:Nn \__enumext_keyans_multi_set_vskip:
958   {
959     \skip_set:Nn \l__enumext_multicols_above_v_skip
960       {
961         \l__enumext_topsep_v_skip
962       }
963     \skip_set:Nn \l__enumext_multicols_below_v_skip
964       {
965         \l__enumext_topsep_v_skip
966       }
967   }
968 \cs_new_protected:Nn \__enumext_keyans_multi_addvspace:
969   {
970     \__enumext_keyans_multi_set_vskip:
971     \mode_if_vertical:T
972       {
973         \skip_add:Nn \l__enumext_multicols_above_v_skip
974           {
975             \skip_use:N \l__enumext_partopsep_v_skip
976           }
977         \skip_add:Nn \l__enumext_multicols_below_v_skip
978           {
979             \skip_use:N \l__enumext_partopsep_v_skip
980           }
981       }
982     \par\nopagebreak
983     \addvspace{ \l__enumext_multicols_above_v_skip }
984   }
```

(*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.
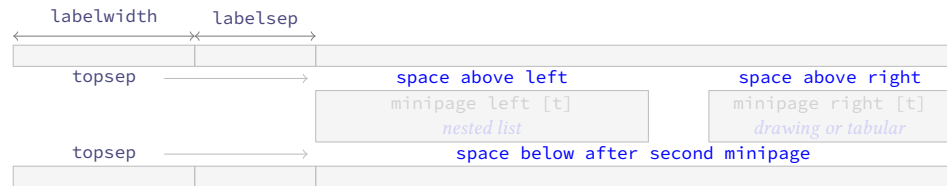


Figure 8: Representation of the minipage spacing adjustment for a nested level.

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.

```
985  \DeclareDocumentEnvironment{__enumext_mini_env*}{ m }
986    {
987      \__enumext_minipage:w [ t ] { #1 }
988        \legacy_if_gset_false:n { @minipage }
989        \vspace { 0pt }
990    }
991    { \__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.

```
992  \cs_new_protected:Nn \__enumext_mini_set_vskip:
993    {
994      \int_compare:nNnTF
995        { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } > { 1 }
996        {
```

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.

```
997          \skip_if_eq:nnTF
998            { \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip } } { \c_zero_skip }
999            {
1000              \skip_set:Nn \l__enumext_minipage_left_skip
1001                {
1002                  -0.150\box_dp:N \strutbox
1003                }
1004              \skip_set:Nn \l__enumext_minipage_right_skip
1005                {
1006                  0.695\box_dp:N \strutbox
1007                }
```

```
1008              \skip_set:Nn \l__enumext_minipage_after_skip
1009                  {
1010                      \box_dp:N \strutbox
1011                  }
1012              \__enumext_zero_parsep:
1013          }
1014          {
1015              \skip_set:Nn \l__enumext_minipage_left_skip
1016                  {
1017                      \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
1018                  }
1019              \skip_set:Nn \l__enumext_minipage_right_skip
1020                  {
1021                      0.695\box_dp:N \strutbox
1022                  }
1023              \skip_set:Nn \l__enumext_minipage_after_skip
1024                  {
1025                      1.85\box_dp:N \strutbox
1026                      + \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
1027                  }
1028          }
1029      }
1030      {
```

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.

```
1031          \skip_if_eq:nnTF
1032              { \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip } } { \c_zero_skip }
1033              {
1034                  \skip_set:Nn \l__enumext_minipage_left_skip
1035                      {
1036                          0.5\box_dp:N \strutbox
1037                          - \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
1038                      }
1039                  \skip_set:Nn \l__enumext_minipage_right_skip
1040                      {
1041                          \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
1042                      }
1043                  \skip_set:Nn \l__enumext_minipage_after_skip
1044                      {
1045                          1.6\box_dp:N \strutbox
1046                      }
1047              }
1048              {
1049                  \skip_set:Nn \l__enumext_minipage_left_skip
1050                      {
1051                          0.5875\box_dp:N \strutbox
1052                          - \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
1053                      }
1054                  \skip_set:Nn \l__enumext_minipage_right_skip
1055                      {
1056                          + \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
1057                          + \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
1058                      }
1059                  \skip_set:Nn \l__enumext_minipage_after_skip
1060                      {
1061                          0.325\box_dp:N \strutbox
1062                          + \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
1063                      }
1064              }
1065          }
1066      }
```

(*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.

```
1067  \cs_new_protected:Nn \__enumext_zero_parsep:
1068      {
```

```
1069        \int_case:nn { \l__enumext_level_int }
1070          {
1071            { 2 }{
1072                  \skip_if_eq:nnF { \l__enumext_parsep_i_skip } { \c_zero_skip }
1073                    {
1074                      \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
1075                    }
1076              }
1077            { 3 }{
1078                  \skip_if_eq:nnF { \l__enumext_parsep_ii_skip } { \c_zero_skip }
1079                    {
1080                      \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
1081                    }
1082              }
1083            { 4 }{
1084                  \skip_if_eq:nnF { \l__enumext_parsep_iii_skip } { \c_zero_skip }
1085                    {
1086                      \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
1087                    }
1088              }
1089          }
1090      }
```

(*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*.

```
1091 \cs_new_protected:Nn \__enumext_mini_addvspace:
1092   {
1093     \__enumext_mini_set_vskip:
1094     \mode_if_vertical:T
1095       {
1096         \skip_add:Nn \l__enumext_minipage_left_skip
1097           {
1098             \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
1099           }
1100         \skip_add:Nn \l__enumext_minipage_after_skip
1101           {
1102             \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
1103           }
1104       }
1105     \par\nopagebreak
1106     \addvspace { \l__enumext_minipage_left_skip }
1107   }
```

(*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.

```
1108 \cs_new_protected:Nn \__enumext_keyans_mini_set_vskip:
1109   {
1110     \skip_zero_new:N \l__enumext_minipage_after_skip
1111     \skip_zero_new:N \l__enumext_minipage_left_skip
1112     \skip_zero_new:N \l__enumext_minipage_right_skip
1113     \int_compare:nNnTF { \l__enumext_columns_v_int } > { 1 }
1114       {
1115         \skip_if_eq:nnTF { \l__enumext_topsep_v_skip } { \c_zero_skip }
1116           {
1117             \skip_set:Nn \l__enumext_minipage_left_skip { -0.25\box_dp:N \strutbox }
1118             \skip_set:Nn \l__enumext_minipage_right_skip { 0.705\box_dp:N \strutbox }
1119             \skip_set:Nn \l__enumext_minipage_after_skip { \box_dp:N \strutbox }
1120             \skip_if_eq:nnF { \l__enumext_parsep_i_skip } { \c_zero_skip }
1121               {
1122                 \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
1123               }
1124           }
```

```
1125                    {
1126                        \skip_set:Nn \l__enumext_minipage_left_skip
1127                            {
1128                                \skip_use:N \l__enumext_topsep_v_skip
1129                            }
1130                        \skip_set:Nn \l__enumext_minipage_right_skip
1131                            {
1132                                0.705\box_dp:N \strutbox
1133                            }
1134                        \skip_set:Nn \l__enumext_minipage_after_skip
1135                            {
1136                                1.85\box_dp:N \strutbox + \l__enumext_topsep_v_skip
1137                            }
1138                    }
1139                }
1140                {
1141                    \skip_if_eq:nnTF { \l__enumext_topsep_v_skip } { \c_zero_skip }
1142                        {
1143                            \skip_set:Nn \l__enumext_minipage_left_skip
1144                                {
1145                                    0.5\box_dp:N \strutbox
1146                                    + \l__enumext_partopsep_v_skip
1147                                }
1148                            \skip_set:Nn \l__enumext_minipage_right_skip
1149                                {
1150                                    \l__enumext_partopsep_v_skip
1151                                }
1152                            \skip_set:Nn \l__enumext_minipage_after_skip { 1.6\box_dp:N \strutbox }
1153                        }
1154                        {
1155                            \skip_set:Nn \l__enumext_minipage_left_skip
1156                                {
1157                                    0.5875\box_dp:N \strutbox - \l__enumext_partopsep_v_skip
1158                                }
1159                            \skip_set:Nn \l__enumext_minipage_right_skip
1160                                {
1161                                    \l__enumext_topsep_v_skip + \l__enumext_partopsep_v_skip
1162                                }
1163                            \skip_set:Nn \l__enumext_minipage_after_skip
1164                                {
1165                                    0.325\box_dp:N \strutbox + \l__enumext_topsep_v_skip
1166                                }
1167                        }
1168                }
1169        }
```

(*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`.

```
1170 \cs_new_protected:Nn \__enumext_keyans_mini_addvspace:
1171    {
1172        \__enumext_keyans_mini_set_vskip:
1173        \mode_if_vertical:T
1174            {
1175                \skip_add:Nn \l__enumext_minipage_left_skip
1176                    {
1177                        \l__enumext_partopsep_v_skip
1178                    }
1179                \skip_add:Nn \l__enumext_minipage_after_skip
1180                    {
1181                        \l__enumext_partopsep_v_skip
1182                    }
1183            }
1184        \par\nopagebreak
1185        \addvspace { \l__enumext_minipage_left_skip }
1186    }
```

(*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*`.

```
1187 \cs_new_protected:Nn \__enumext_mini_set_vskip_vii:
1188   {
1189     \skip_zero_new:N \l__enumext_minipage_left_skip
1190     \skip_gzero_new:N \g__enumext_minipage_right_skip
1191     \skip_gzero_new:N \g__enumext_minipage_after_skip
1192     \skip_if_eq:nnTF { \l__enumext_topsep_vii_skip } { \c_zero_skip }
1193       {
1194         \skip_set:Nn \l__enumext_minipage_left_skip { 0.5\box_dp:N \strutbox }
1195         \skip_gset:Nn \g__enumext_minipage_right_skip { 0.325\box_dp:N \strutbox }
1196       }
1197       {
1198         \skip_set:Nn \l__enumext_minipage_left_skip { 0.5875\box_dp:N \strutbox }
1199         \skip_gset:Nn \g__enumext_minipage_right_skip
1200           {
1201             \l__enumext_topsep_vii_skip
1202           }
1203         \skip_gset:Nn \g__enumext_minipage_after_skip
1204           {
1205             0.325\box_dp:N \strutbox + \l__enumext_topsep_vii_skip
1206           }
1207       }
1208   }
1209 \cs_new_protected:Nn \__enumext_mini_set_vskip_viii:
1210   {
1211     \skip_zero_new:N \l__enumext_minipage_after_skip
1212     \skip_zero_new:N \l__enumext_minipage_left_skip
1213     \skip_zero_new:N \l__enumext_minipage_right_skip
1214     \skip_if_eq:nnTF { \l__enumext_topsep_viii_skip } { \c_zero_skip }
1215       {
1216         \skip_set:Nn \l__enumext_minipage_left_skip
1217           {
1218             0.5\box_dp:N \strutbox
1219           }
1220         \skip_set:Nn \l__enumext_minipage_right_skip
1221           {
1222             \l__enumext_partopsep_viii_skip
1223           }
1224         \skip_set:Nn \l__enumext_minipage_after_skip
1225           {
1226             1.6\box_dp:N \strutbox
1227           }
1228       }
1229       {
1230         \skip_set:Nn \l__enumext_minipage_left_skip
1231           {
1232             0.5875\box_dp:N \strutbox
1233           }
1234         \skip_set:Nn \l__enumext_minipage_right_skip
1235           {
1236             \l__enumext_topsep_viii_skip
1237           }
1238         \skip_set:Nn \l__enumext_minipage_after_skip
1239           {
1240             0.325\box_dp:N \strutbox + \l__enumext_topsep_viii_skip
1241           }
1242       }
1243   }
```

(*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.

```
1244 \cs_new_protected:Nn \__enumext_mini_addvspace_vii:
```

```
1245    {
1246      \__enumext_mini_set_vskip_vii:
1247      \par\nopagebreak
1248      \addvspace { \l__enumext_minipage_left_skip }
1249    }
1250  \cs_new_protected:Nn \__enumext_mini_addvspace_viii:
1251    {
1252      \__enumext_mini_set_vskip_viii:
1253      \par\nopagebreak
1254      \addvspace { \l__enumext_minipage_left_skip }
1255    }
```

(*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.

```
1256  \NewDocumentCommand \miniright { s }
1257    {
1258      \int_compare:nNnT { \l__enumext_keyans_pic_level_int } = { 1 }
1259        {
1260          \msg_error:nnn { enumext } { wrong-miniright-place }
1261        }
1262      \int_compare:nNnT { \l__enumext_level_int } = { 0 }
1263        {
1264          \msg_error:nnn { enumext } { wrong-miniright-place }
1265        }
1266      \int_compare:nNnTF { \l__enumext_keyans_level_int } = { 1 }
1267        {
1268          \__enumext_keyans_mini_right_cmd:n {#1}
1269        }
1270        { \__enumext_mini_right_cmd:n {#1} }
1271    }
```

(*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.

```
1272  \cs_new_protected:Npn \__enumext_mini_right_cmd:n #1
1273    {
1274      \dim_compare:nNnTF
1275        { \dim_use:c { l__enumext_minipage_right_ \__enumext_level: _dim } } > { \c_zero_dim }
1276        {
1277          \__enumext_multicols_stop:
1278          \end{__enumext_mini_env*}
1279          \hfill
1280          \begin{__enumext_mini_env*}
1281            { \dim_use:c { l__enumext_minipage_right_ \__enumext_level: _dim } }
1282          \par\addvspace { \l__enumext_minipage_right_skip }
1283          \bool_if:nF {#1}
1284            {
1285              \centering
1286            }
1287          \int_gzero:N \g__enumext_minipage_stat_int
1288        }
1289        { \msg_error:nnn { enumext } { wrong-miniright-use } }
1290    }
```

(*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.

```
1291 \cs_new_protected:Npn \__enumext_keyans_mini_right_cmd:n #1
1292   {
1293     \dim_compare:nNnTF { \l__enumext_minipage_right_v_dim } > { \c_zero_dim }
1294       {
1295         \__enumext_keyans_multicols_stop:
1296         \end{__enumext_mini_env*}
1297         \hfill
1298         \begin{__enumext_mini_env*}{ \l__enumext_minipage_right_v_dim }
1299           \par\addvspace { \l__enumext_minipage_right_skip }
1300           \bool_if:nF {#1}
1301             {
1302               \centering
1303             }
1304           \int_gzero:N \g__enumext_minipage_stat_int
1305       }
1306     { \msg_error:nnn { enumext } { wrong-miniright-use } }
1307   }
```

(*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.

```
1308 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1309   {
1310     \keys_define:nn { enumext / #1 }
1311       {
1312         above  .skip_set:c = { l__enumext_vspace_above_#2_skip },
1313         above  .value_required:n = true,
1314         above* .code:n     = \bool_set_true:c { l__enumext_vspace_a_star_#2_bool }
1315                              \keys_set:nn { enumext / #1 } { above = {##1} },
1316         above* .value_required:n = true,
1317         below  .skip_set:c = { l__enumext_vspace_below_#2_skip },
1318         below  .value_required:n = true,
1319         below* .code:n     = \bool_set_true:c { l__enumext_vspace_b_star_#2_bool }
1320                              \keys_set:nn { enumext / #1 } { below = {##1} },
1321         below* .value_required:n = true,
1322       }
1323   }
1324 \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.

```
1325 \cs_new_protected:Nn \__enumext_vspace_above:
1326   {
1327     \skip_if_eq:nnF
1328       { \skip_use:c { l__enumext_vspace_above_ \__enumext_level: _skip } } { \c_zero_skip }
1329       {
1330         \bool_if:cTF { l__enumext_vspace_a_star_ \__enumext_level: _bool }
1331           {
1332             \vspace*{ \skip_use:c { l__enumext_vspace_above_ \__enumext_level: _skip } }
1333           }
1334           {
1335             \vspace { \skip_use:c { l__enumext_vspace_above_ \__enumext_level: _skip } }
1336           }
1337       }
1338   }
```

(*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.

```
1339 \cs_new_protected:Nn \__enumext_vspace_below:
1340   {
1341     \skip_if_eq:nnF
1342       { \skip_use:c { l__enumext_vspace_below_ \__enumext_level: _skip } } { \c_zero_skip }
1343       {
1344         \bool_if:cTF { l__enumext_vspace_b_star_ \__enumext_level: _bool }
1345           {
1346             \vspace*{ \skip_use:c { l__enumext_vspace_below_ \__enumext_level: _skip } }
1347           }
1348           {
1349             \vspace { \skip_use:c { l__enumext_vspace_below_ \__enumext_level: _skip } }
1350           }
1351       }
1352   }
```

(*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.

```
1353 \cs_new_protected:Nn \__enumext_vspace_above_v:
1354   {
1355     \skip_if_eq:nnF { \l__enumext_vspace_above_v_skip } { \c_zero_skip }
1356       {
1357         \bool_if:NTF \l__enumext_vspace_a_star_v_bool
1358           {
1359             \vspace*{ \l__enumext_vspace_above_v_skip }
1360           }
1361           { \vspace { \l__enumext_vspace_above_v_skip } }
1362       }
1363   }
```

(*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.

```
1364 \cs_new_protected:Nn \__enumext_vspace_below_v:
1365   {
1366     \skip_if_eq:nnF { \l__enumext_vspace_below_v_skip } { \c_zero_skip }
1367       {
1368         \bool_if:NTF \l__enumext_vspace_b_star_v_bool
1369           {
1370             \vspace*{ \l__enumext_vspace_below_v_skip }
1371           }
1372           { \vspace { \l__enumext_vspace_below_v_skip } }
1373       }
1374   }
```

(*End of definition for* \_\_enumext_vspace_below_v:*.*)

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

\_\_enumext_vspace_above_vii:
\_\_enumext_vspace_above_viii:

The functions \_\_enumext_vspace_above_vii: and \_\_enumext_vspace_above_viii: apply the *vertical space above* the enumext* and keyans* environments set by the above and above* keys.

```
1375 \cs_new_protected:Nn \__enumext_vspace_above_vii:
1376   {
1377     \skip_if_eq:nnF { \l__enumext_vspace_above_vii_skip } { \c_zero_skip }
1378       {
1379         \bool_if:NTF \l__enumext_vspace_a_star_vii_bool
1380           {
1381             \vspace*{ \l__enumext_vspace_above_vii_skip }
1382           }
1383           { \vspace { \l__enumext_vspace_above_vii_skip } }
1384       }
1385   }
1386 \cs_new_protected:Nn \__enumext_vspace_above_viii:
1387   {
1388     \skip_if_eq:nnF { \l__enumext_vspace_above_viii_skip } { \c_zero_skip }
```

```
1389        {
1390          \bool_if:NTF \l__enumext_vspace_a_star_viii_bool
1391            {
1392              \vspace*{ \l__enumext_vspace_above_viii_skip }
1393            }
1394            { \vspace { \l__enumext_vspace_above_viii_skip } }
1395        }
1396    }
```

(*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.

```
1397 \cs_new_protected:Nn \__enumext_vspace_below_vii:
1398    {
1399      \skip_if_eq:nnF { \l__enumext_vspace_below_vii_skip } { \c_zero_skip }
1400        {
1401          \bool_if:NTF \l__enumext_vspace_b_star_vii_bool
1402            {
1403              \vspace*{ \l__enumext_vspace_below_vii_skip }
1404            }
1405            { \vspace { \l__enumext_vspace_below_vii_skip } }
1406        }
1407    }
1408 \cs_new_protected:Nn \__enumext_vspace_below_viii:
1409    {
1410      \skip_if_eq:nnF { \l__enumext_vspace_below_viii_skip } { \c_zero_skip }
1411        {
1412          \bool_if:NTF \l__enumext_vspace_b_star_viii_bool
1413            {
1414              \vspace*{ \l__enumext_vspace_below_viii_skip }
1415            }
1416            { \vspace { \l__enumext_vspace_below_viii_skip } }
1417        }
1418    }
```

(*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*.

```
1419 \cs_set_protected:Npn \__enumext_tmp:n #1
1420    {
1421      \keys_define:nn { enumext / #1 }
1422        {
1423          series  .str_set:N = \l__enumext_series_str,
1424          series  .value_required:n = true,
1425          resume  .code:n = \__enumext_resume_series:n {##1},
1426          resume* .code:n = \__enumext_resume_starred:,
1427          resume* .value_forbidden:n = true,
1428        }
1429    }
1430 \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.

```
1431 \cs_new:Npn \__enumext_filter_series:n #1
1432    {
1433      \use:e
1434        {
1435          \keyval_parse:NNn
1436            \__enumext_filter_series_key:n
1437            \__enumext_filter_series_pair:nn {#1}
1438        }
1439    }
```

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.

```
1440  \cs_new:Npn \__enumext_filter_series_key:n #1
1441    {
1442      \str_case:nnF {#1}
1443        {
1444          { resume } {}
1445          { resume* } {}
1446        }
1447        { , { \exp_not:n {#1} } }
1448    }
```

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.

```
1449  \cs_new:Npn \__enumext_filter_series_pair:nn #1#2
1450    {
1451      \str_case:nnF {#1}
1452        {
1453          { series } {}
1454          { resume } {}
1455          { start } {}
1456          { save-ans } {}
1457          { save-key } {}
1458        }
1459        { , { \exp_not:n {#1} } = { \exp_not:n {#2} } }
1460    }
```

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

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

🎣 The value of boolean variable `\l__enumext_resume_active_bool` is set to true by the function `\__enumext_-resume_counter:n` which is used by the keys resume and resume*, in this case we must Make sure it is set to false so that it does not overwrite the default filtered ⟨keys⟩. This function is passed to the function `\__enumext_parse_-keys:n` in the enumext environment definition (§10.32) and to the function `\__enumext_parse_keys_vii:n` in the enumext* environment definition (§10.35).

```
1461  \cs_new_protected:Npn \__enumext_parse_series:n #1
1462    {
1463      \str_if_empty:NTF \l__enumext_series_str
1464        {
1465          \bool_if:NF \l__enumext_resume_active_bool
1466            {
1467              \__enumext_resume_last:n {#1}
1468            }
1469        }
1470        {
1471          \tl_gclear_new:c { g__enumext_series_ \l__enumext_series_str _tl }
1472          \tl_gset:ce { g__enumext_series_ \l__enumext_series_str _tl }
1473            { \__enumext_filter_series:n {#1} }
1474          \int_if_exist:cF { g__enumext_series_ \l__enumext_series_str _int }
1475            {
1476              \int_new:c { g__enumext_series_ \l__enumext_series_str _int }
1477            }
1478        }
1479    }
```

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.

```
1480  \cs_new_protected:Npn \__enumext_resume_last:n #1
1481    {
1482      \bool_if:NT \l__enumext_standar_level_one_bool
1483        {
1484          \tl_gclear:N \g__enumext_standar_series_tl
1485          \tl_gset:Ne \g__enumext_standar_series_tl { \__enumext_filter_series:n {#1} }
```

```
1486              }
1487          \bool_if:NT \l__enumext_starred_level_one_bool
1488              {
1489                  \tl_gclear:N \g__enumext_starred_series_tl
1490                  \tl_gset:Ne \g__enumext_starred_series_tl { \__enumext_filter_series:n {#1} }
1491              }
1492      }
```

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

### 10.21.2 Internal function to save counter value

\__enumext_resume_save_counter:

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

💣 The variables \l__enumext_series_str and \l__enumext__resume_name_tl contain the same {⟨*series name*⟩} but are executed at different moments, the integer variable with \l__enumext_series_str sets the value when execute series={⟨*series name*⟩} and the integer variable with \l__enumext__resume_name_tl sets the subsequent values when use resume={⟨*series name*⟩}. This function is passed to the enumext environment definition (§10.32) and the enumext* environment definition (§10.35).

```
1493  \cs_new_protected:Nn \__enumext_resume_save_counter:
1494      {
1495          \bool_if:NT \g__enumext_standar_bool
1496              {
1497                  \tl_if_empty:NF \l__enumext_series_str
1498                      {
1499                          \int_gset_eq:cN
1500                              { g__enumext_series_ \l__enumext_series_str _int } \value{enumXi}
1501                      }
1502                  \tl_if_empty:NTF \l__enumext_resume_name_tl
1503                      {
1504                          \str_if_empty:NT \l__enumext_series_str
1505                              {
1506                                  \int_gset_eq:NN \g__enumext_resume_int \value{enumXi}
1507                              }
1508                      }
1509                      {
1510                          \int_if_exist:cT { g__enumext_series_ \l__enumext_resume_name_tl _int }
1511                              {
1512                                  \int_gset_eq:cN
1513                                      { g__enumext_series_ \l__enumext_resume_name_tl _int } \value{enumXi}
1514                              }
1515                      }
1516                  \int_if_exist:cT { g__enumext_resume_ \l__enumext_store_name_tl _int }
1517                      {
1518                          \int_gset_eq:cN
1519                              { g__enumext_resume_ \l__enumext_store_name_tl _int } \value{enumXi}
1520                      }
1521              }
1522          \bool_if:NT \g__enumext_starred_bool
1523              {
1524                  \tl_if_empty:NF \l__enumext_series_str
1525                      {
1526                          \int_gset_eq:cN
1527                              { g__enumext_series_ \l__enumext_series_str _int } \value{enumXvii}
1528                      }
1529                  \tl_if_empty:NTF \l__enumext_resume_name_tl
1530                      {
1531                          \str_if_empty:NT \l__enumext_series_str
1532                              {
1533                                  \int_gset_eq:NN \g__enumext_resume_vii_int \value{enumXvii}
1534                              }
1535                      }
1536                      {
1537                          \int_if_exist:cT { g__enumext_series_ \l__enumext_resume_name_tl _int }
1538                              {
1539                                  \int_gset_eq:cN
1540                                      { g__enumext_series_ \l__enumext_resume_name_tl _int } \value{enumXvii}
1541                              }
```

```
1542              }
1543          \int_if_exist:cT { g__enumext_resume_ \l__enumext_store_name_tl _int }
1544              {
1545                \int_gset_eq:cN
1546                  { g__enumext_resume_ \l__enumext_store_name_tl _int } \value{enumXvii}
1547              }
1548          }
1549      }
```

(*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.

```
1550  \cs_new_protected:Npn \__enumext_resume_series:n #1
1551    {
1552      \tl_if_empty:nTF {#1}
1553        {
1554          \__enumext_resume_counter:n { }
1555        }
1556        {
1557          \tl_if_exist:cTF { g__enumext_series_ \tl_to_str:n {#1} _tl }
1558            {
1559              \__enumext_resume_counter:n {#1}
1560              \bool_if:NT \g__enumext_standar_bool
1561                {
1562                  \keys_set:nv { enumext / level-1 }
1563                    { g__enumext_series_ \tl_to_str:n {#1} _tl }
1564                }
1565              \bool_if:NT \g__enumext_starred_bool
1566                {
1567                  \keys_set:nv { enumext / enumext* }
1568                    { g__enumext_series_ \tl_to_str:n {#1} _tl }
1569                }
1570            }
1571            {
1572              \bool_if:NT \g__enumext_standar_bool
1573                {
1574                  \msg_error:nnn { enumext } { unknown-series } {#1}
1575                }
1576              \bool_if:NT \g__enumext_starred_bool
1577                {
1578                  \msg_error:nnn { enumext } { unknown-series } {#1}
1579                }
1580            }
1581        }
1582    }
```

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

\__enumext_resume_counter:n
\__enumext_resume_counter:
\__enumext_resume_counter_series:
\__enumext_resume_counter_save_ans:

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

```
1583  \cs_new_protected:Npn \__enumext_resume_counter:n #1
1584    {
1585      \bool_set_true:N \l__enumext_resume_active_bool
1586      \tl_set:Nn \l__enumext_resume_name_tl {#1}
1587      \tl_if_empty:NTF \l__enumext_resume_name_tl
1588        {
1589          \__enumext_resume_counter:
1590        }
```

```
1591          {
1592              \__enumext_resume_counter_series:
1593          }
1594      \__enumext_resume_counter_save_ans:
1595    }
```

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.

```
1596  \cs_new_protected:Nn \__enumext_resume_counter:
1597    {
1598      \bool_if:NT \g__enumext_standar_bool
1599        {
1600          \int_gincr:N \g__enumext_resume_int
1601          \int_set_eq:NN \l__enumext_start_i_int \g__enumext_resume_int
1602        }
1603      \bool_if:NT \g__enumext_starred_bool
1604        {
1605          \int_gincr:N \g__enumext_resume_vii_int
1606          \int_set_eq:NN \l__enumext_start_vii_int \g__enumext_resume_vii_int
1607        }
1608    }
```

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.

```
1609  \cs_new_protected:Nn \__enumext_resume_counter_series:
1610    {
1611      \bool_if:NT \g__enumext_standar_bool
1612        {
1613          \int_set:Nn \l__enumext_start_i_int
1614            {
1615              \int_use:c { g__enumext_series_ \l__enumext_resume_name_tl _int } + 1
1616            }
1617        }
1618      \bool_if:NT \g__enumext_starred_bool
1619        {
1620          \int_set:Nn \l__enumext_start_vii_int
1621            {
1622              \int_use:c { g__enumext_series_ \l__enumext_resume_name_tl _int } + 1
1623            }
1624        }
1625    }
```

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.

```
1626  \cs_new_protected:Nn \__enumext_resume_counter_save_ans:
1627    {
1628      \bool_lazy_and:nnT
1629        { \bool_if_p:N \l__enumext_standar_level_one_bool }
1630        { \bool_if_p:N \l__enumext_store_active_bool }
1631        {
1632          \int_set:Nn \l__enumext_start_i_int
1633            {
1634              \int_use:c { g__enumext_resume_ \l__enumext_store_name_tl _int } + 1
1635            }
1636        }
1637      \bool_lazy_and:nnT
1638        { \bool_if_p:N \l__enumext_starred_level_one_bool }
1639        { \bool_if_p:N \l__enumext_store_active_bool }
1640        {
1641          \int_set:Nn \l__enumext_start_vii_int
1642            {
1643              \int_use:c { g__enumext_resume_ \l__enumext_store_name_tl _int } + 1
1644            }
1645        }
1646    }
```

(*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.

```
1647 \cs_new_protected:Nn \__enumext_resume_starred:
1648   {
1649     \bool_if:NT \g__enumext_standar_bool
1650       {
1651         \tl_if_empty:NF \g__enumext_standar_series_tl
1652           {
1653             \__enumext_resume_counter:n { }
1654             \keys_set:nV { enumext / level-1 } \g__enumext_standar_series_tl
1655           }
1656       }
1657     \bool_if:NT \g__enumext_starred_bool
1658       {
1659         \tl_if_empty:NF \g__enumext_starred_series_tl
1660           {
1661             \__enumext_resume_counter:n { }
1662             \keys_set:nV { enumext / enumext* } \g__enumext_starred_series_tl
1663           }
1664       }
1665   }
```

(*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*.

```
1666 \cs_set_protected:Npn \__enumext_tmp:n #1
1667   {
1668     \keys_define:nn { enumext / #1 }
1669       {
1670         save-ans .code:n = \__enumext_storing_set:n {##1},
1671         save-ans .value_required:n = true,
1672       }
1673   }
1674 \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.

```
1675 \cs_new_protected:Npn \__enumext_storing_set:n #1
1676   {
1677     \tl_set:Ne \l__enumext_store_name_tl {#1}
1678     \tl_if_empty:NTF \l__enumext_store_name_tl
1679       {
1680         \bool_if:NT \l__enumext_standar_level_one_bool
1681           {
1682             \msg_error:nnn { enumext } { save-ans-empty } { enumext }
1683           }
1684         \bool_if:NT \l__enumext_starred_level_one_bool
1685           {
1686             \msg_error:nnn { enumext } { save-ans-empty } { enumext* }
1687           }
1688       }
1689       {
1690         \bool_if:NT \l__enumext_standar_level_one_bool
1691           {
1692             \msg_note:nnnV
1693               { enumext } { save-ans-ok } { enumext } \l__enumext_store_name_tl
1694             \__enumext_storing_exec:
```

```
1695            }
1696        \bool_if:NT \l__enumext_starred_level_one_bool
1697          {
1698            \msg_note:nnnV
1699              { enumext } { save-ans-ok } { enumext* } \l__enumext_store_name_tl
1700            \__enumext_storing_exec:
1701          }
1702      }
1703    }
```

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*`.

```
1704 \cs_new_protected:Nn \__enumext_storing_exec:
1705   {
1706     \bool_set_true:N \l__enumext_store_active_bool
1707     \bool_set_true:N \l__enumext_store_ans_bool
1708     \prop_if_exist:cF { g__enumext_ \l__enumext_store_name_tl _prop }
1709       {
1710         \prop_new:c { g__enumext_ \l__enumext_store_name_tl _prop }
1711       }
1712     \seq_if_exist:cF { g__enumext_ \l__enumext_store_name_tl _seq }
1713       {
1714         \seq_new:c { g__enumext_ \l__enumext_store_name_tl _seq }
1715       }
1716     \int_if_exist:cF { g__enumext_resume_ \l__enumext_store_name_tl _int }
1717       {
1718         \int_new:c { g__enumext_resume_ \l__enumext_store_name_tl _int }
1719       }
1720   }
```

(*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.

```
1721 \cs_set_protected:Npn \__enumext_tmp:n #1
1722   {
1723     \keys_define:nn { enumext / #1 }
1724       {
1725         check-ans .bool_set:N = \l__enumext_check_ans_bool,
1726         check-ans .initial:n  = false,
1727         no-store  .code:n = {
1728                             \bool_set_false:N \l__enumext_store_ans_bool
1729                             \bool_set_false:N \l__enumext_check_ans_bool
1730                           },
1731         no-store  .value_forbidden:n = true,
1732       }
1733   }
1734 \clist_map_inline:nn
1735   {
1736     level-1, level-2, level-3, level-4, enumext*
1737   }
1738   { \__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.

```
1739 \cs_new_protected:Nn \__enumext_check_ans_set:
1740   {
1741     \int_case:nn { \l__enumext_level_int }
1742       {
1743         { 1 }{
1744               \bool_lazy_all:nT
1745                 {
1746                   { \bool_if_p:N \g__enumext_starred_bool }
1747                   { \int_compare_p:nNn { \l__enumext_level_h_int } = { 1 } }
1748                 }
1749                 {
1750                   \int_gdecr:N \g__enumext_count_item_number_int
1751                   \typeout{ENUMEXT ~ STANDAR ~ NEEEEEEEEEEEESTED}
1752                 }
1753             }
1754         { 2 }{
1755               \int_gdecr:N \g__enumext_count_item_number_int
1756             }
1757         { 3 }{
1758               \int_gdecr:N \g__enumext_count_item_number_int
1759             }
1760         { 4 }{
1761               \int_gdecr:N \g__enumext_count_item_number_int
1762             }
1763       }
1764     \int_case:nn { \l__enumext_level_h_int }
1765       {
1766         { 1 }{
1767               \bool_if:NT \g__enumext_standar_bool
1768                 {
1769                   \int_gdecr:N \g__enumext_count_item_number_int
1770                   \typeout{ENUMEXT ~ STARRED ~ NEEEEEEEEEEEESTED}
1771                 }
1772             }
1773       }
1774   }
```

(*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.

```
1775 \cs_new_protected:Nn \__enumext_check_ans_exec:
1776   {
1777     \bool_if:NT \l__enumext_check_ans_bool
1778       {
1779         \__enumext_check_ans_set:
1780       }
1781   }
```

(*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.

```
1782 \cs_new_protected:Nn \__enumext_check_ans_show:
1783   {
1784     \int_compare:nNnTF
1785       { \g__enumext_count_item_number_int } = { \g__enumext_count_item_anskey_int }
1786       {
1787         \msg_term:nnV { enumext } { items-same-answer } \g__enumext_store_name_tl
1788       }
1789       {
1790         \msg_warning:nnV { enumext } { item-different-answer } \g__enumext_store_name_tl
1791       }
1792     \int_gzero:N \g__enumext_count_item_number_int
1793     \int_gzero:N \g__enumext_count_item_anskey_int
1794   }
```

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

### 10.23.3   Check for \item* and \anspic* commands

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

```
1795 \cs_new_protected:Npn \__enumext_check_starred_cmd:nn #1 #2
1796   {
1797     \tl_if_empty:NTF \g__enumext_check_ans_item_tl
1798       {
1799         \msg_warning:nnnn { enumext } { missing-starred }{ #1 }{ #2 }
1800       }
1801       { \tl_gclear:N \g__enumext_check_ans_item_tl }
1802   }
```

(*End of definition for* \__enumext_check_starred_cmd:nn.)

## 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*.

```
1803 \cs_set_protected:Npn \__enumext_tmp:n #1
1804   {
1805     \keys_define:nn { enumext / #1 }
1806       {
1807         wrap-ans   .cs_set_protected:Np = \__enumext_anskey_wrapper:n ##1,
1808         wrap-ans   .initial:n = \fbox{##1},
1809         wrap-ans   .value_required:n = true,
1810         wrap-opt   .cs_set_protected:Np = \__enumext_keyans_wrapper_opt:n ##1,
1811         wrap-opt   .initial:n = [{##1}],
1812         wrap-opt   .value_required:n = true,
1813         save-sep   .tl_set:N  = \l__enumext_store_keyans_item_opt_sep_tl,
1814         save-sep   .initial:n = {, ~ },
1815         save-sep   .value_required:n = true,
1816         mark-ans   .tl_set:N  = \l__enumext_mark_answer_sym_tl,
1817         mark-ans   .initial:n = \textasteriskcentered,
1818         mark-ans   .value_required:n = true,
1819         mark-pos   .choice:,
1820         mark-pos  / left   .code:n = \str_set:Nn \l__enumext_mark_position_str { l },
1821         mark-pos  / right  .code:n = \str_set:Nn \l__enumext_mark_position_str { r },
1822         mark-pos   .initial:n = right,
1823         mark-pos   .value_required:n = true,
1824         show-ans   .bool_set:N = \l__enumext_show_answer_bool,
1825         show-ans   .initial:n  = false,
```

```
1826          show-ans    .value_required:n = true,
1827          show-pos    .bool_set:N = \l__enumext_show_position_bool,
1828          show-pos    .initial:n  = false,
1829          show-pos    .value_required:n = true,
1830          mark-ref    .tl_set:N   = \l__enumext_mark_ref_sym_tl,
1831          mark-ref    .initial:n  = \textasteriskcentered,
1832          mark-ref    .value_required:n = true,
1833          save-ref    .bool_set:N = \l__enumext_store_ref_key_bool,
1834          save-ref    .initial:n  = false,
1835          save-ref    .value_required:n = true,
1836        }
1837    }
1838 \clist_map_inline:nn { level-1, enumext* } { \__enumext_tmp:n {#1} }
```

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

mark-pos  
show-ans  
show-pos

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

```
1839 \cs_set_protected:Npn \__enumext_tmp:n #1
1840   {
1841     \keys_define:nn { enumext / #1 }
1842       {
1843         mark-pos .choice:,
1844         mark-pos / left   .code:n = \str_set:Nn \l__enumext_mark_position_str { l },
1845         mark-pos / right  .code:n = \str_set:Nn \l__enumext_mark_position_str { r },
1846         mark-pos .initial:n = right,
1847         mark-pos .value_required:n  = true,
1848         show-ans .bool_set:N = \l__enumext_show_answer_bool,
1849         show-ans .initial:n  = false,
1850         show-ans .value_required:n = true,
1851         show-pos .bool_set:N = \l__enumext_show_position_bool,
1852         show-pos .initial:n  = false,
1853         show-pos .value_required:n = true,
1854       }
1855   }
1856 \clist_map_inline:nn { keyans, keyans* } { \__enumext_tmp:n {#1} }
```

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

columns*  
columns-sep*

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

```
1857 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1858   {
1859     \keys_define:nn { enumext / #1 }
1860       {
1861         columns*     .code:n = \bool_set_true:c { l__enumext_store_columns_#2_bool }
1862                               \int_set:cn { l__enumext_store_columns_#2_int } {##1}
1863                               \tl_put_right:ce { l__enumext_store_opt_#2_tl }
1864                                 {
1865                                   columns = \exp_not:v { l__enumext_store_columns_#2_int },
1866                                 },
1867         columns*     .value_required:n  = true,
1868         columns-sep* .code:n = \bool_set_true:c { l__enumext_store_columns_sep_#2_bool }
1869                               \dim_set:cn { l__enumext_store_columns_sep_#2_dim } {##1}
1870                               \tl_put_right:ce { l__enumext_store_opt_#2_tl }
1871                                 {
1872                                   columns-sep = \exp_not:v { l__enumext_store_columns_sep_#2_dim
1873                                 },
1874         columns-sep* .value_required:n  = true,
1875       }
1876   }
1877 \clist_map_inline:nn
1878   {
1879     {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {enumext*}{vii}
1880   }
1881   { \__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.

```
1882 \cs_generate_variant:Nn \prop_gput_if_not_in:Nnn { cen }
1883 \cs_new_protected:Npn \__enumext_store_addto_prop:n #1
1884   {
1885     \prop_gput_if_not_in:cen { g__enumext_ \l__enumext_store_name_tl _prop }
1886       {
1887         \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } + 1 }
1888       }
1889       { #1 }
1890   }
1891 \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.

```
1892 \cs_new_protected:Npn \__enumext_store_addto_seq:n #1
1893   {
1894     \seq_gput_right:cn { g__enumext_ \l__enumext_store_name_tl _seq } { #1 }
1895   }
1896 \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.

```
1897 \cs_new_protected:Nn \__enumext_store_level_open:
1898   {
1899     \bool_if:NT \l__enumext_store_ans_bool
1900       {
1901         \tl_if_empty:cTF { l__enumext_store_opt_ \__enumext_level: _tl }
1902           {
1903             \__enumext_store_addto_seq:n
1904               {
1905                 \item \begin{enumext}
1906               }
1907           }
1908           {
1909             \tl_put_left:cn { l__enumext_store_opt_ \__enumext_level: _tl }
1910               {
1911                 \item \begin{enumext} [
1912               }
1913             \tl_put_right:cn { l__enumext_store_opt_ \__enumext_level: _tl }
1914               {
1915                 ]
1916               }
1917             \__enumext_store_addto_seq:v { l__enumext_store_opt_ \__enumext_level: _tl }
1918           }
1919       }
1920   }
1921 \cs_new_protected:Nn \__enumext_store_level_close:
1922   {
1923     \bool_if:NT \l__enumext_store_ans_bool
1924       {
```

```
1925                    \__enumext_store_addto_seq:n { \end{enumext} }
1926                }
1927        }
```

(*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.

```
1928  \cs_new_protected:Nn \__enumext_store_level_open_vii:
1929    {
1930      \bool_if:NT \l__enumext_store_ans_bool
1931        {
1932          \tl_if_empty:NTF \l__enumext_store_opt_vii_tl
1933            {
1934              \__enumext_store_addto_seq:n
1935                {
1936                  \item \mode_leave_vertical:
1937                  \vspace { -\skip_eval:n { \baselineskip + \parsep } }
1938                  \begin{enumext*}[before={\setlength{\topsep}{0pt}},]
1939                }
1940            }
1941            {
1942              \tl_put_left:Nn \l__enumext_store_opt_vii_tl
1943                {
1944                  \item \mode_leave_vertical:
1945                  \vspace { -\skip_eval:n { \baselineskip + \parsep } }
1946                  \begin{enumext*}[before={\setlength{\topsep}{0pt}},
1947                }
1948              \tl_put_right:Nn \l__enumext_store_opt_vii_tl
1949                {
1950                  ]
1951                }
1952              \__enumext_store_addto_seq:V \l__enumext_store_opt_vii_tl
1953            }
1954        }
1955    }
1956  \cs_new_protected:Nn \__enumext_store_level_close_vii:
1957    {
1958      \bool_if:NT \l__enumext_store_ans_bool
1959        {
1960          \__enumext_store_addto_seq:n { \end{enumext*} }
1961        }
1962    }
```

(*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

```
1963  \cs_new_protected:Nn \__enumext_print_keyans_box:NN
1964    {
1965      \mode_leave_vertical:
1966      \skip_horizontal:n { -\dim_use:N #2 }
1967      \makebox[0pt][ r ]
1968        {
1969          \makebox[ \dim_use:N #1 ][ \l__enumext_mark_position_str ]
1970            {
1971              \tl_use:N \l__enumext_mark_answer_sym_tl
1972            }
1973        }
1974      \skip_horizontal:n { \dim_use:N #2 }
1975    }
1976  \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.

```
1977 \keys_define:nn { enumext / anskey }
1978   {
1979     item-sym* .tl_set:N   = \l__enumext_store_item_symbol_tl,
1980     item-sym* .value_required:n = true,
1981     item-pos* .dim_set:N  = \l__enumext_store_item_symbol_sep_dim,
1982     item-pos* .value_required:n = true,
1983     store-brk .bool_set:N = \l__enumext_store_columns_break_bool,
1984     store-brk .default:n  = true,
1985     store-brk .value_forbidden:n = true,
1986   }
```

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*⟩.

```
1987 \NewDocumentCommand \anskey { d() s o +m }
1988   {
1989     \bool_if:NF \l__enumext_store_active_bool
1990       {
1991         \msg_error:nnnn { enumext } { anskey-wrong-place }{ anskey }{ enumext }
1992       }
1993     \int_compare:nNnT { \l__enumext_keyans_level_int } = { 1 }
1994       {
1995         \msg_error:nnnn { enumext } { command-wrong-place }{ anskey }{ keyans }
1996       }
1997     \int_compare:nNnT { \l__enumext_keyans_pic_level_int } = { 1 }
1998       {
1999         \msg_error:nnnn { enumext } { command-wrong-place }{ anskey }{ keyanspic }
2000       }
2001     \group_begin:
2002       \bool_if:NT \l__enumext_store_ans_bool
2003         {
2004           \bool_if:NT \l__enumext_check_ans_bool
2005             {
2006               \int_gincr:N \g__enumext_count_item_anskey_int
2007             }
2008           \__enumext_store_anskey_code:nnnn {#1} {#2} {#3} {#4}
2009         }
2010     \group_end:
2011   }
```

(*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.

```
2012 \cs_new_protected:Npn \__enumext_store_anskey_code:nnnn #1 #2 #3 #4
2013   {
2014     \__enumext_store_addto_prop:n {#4}
2015     \bool_if:NT \l__enumext_store_ref_key_bool
2016       {
2017         \__enumext_store_internal_ref:
2018       }
2019     \__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.

```
2020        \tl_clear:N \l__enumext_store_anskey_arg_tl
2021        \tl_if_novalue:nF {#3}
2022          {
2023            \keys_set:nn { enumext / anskey } {#3}
2024          }
2025        \bool_lazy_and:nnT
2026          { \bool_if_p:N \l__enumext_store_columns_break_bool }
2027          { \bool_not_p:n { \l__enumext_starred_bool } }
2028          {
2029            \tl_put_left:Nn \l__enumext_store_anskey_arg_tl { \columnbreak }
2030          }
2031        \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).

```
2032        \tl_if_novalue:nF {#1}
2033          {
2034            \int_set:Nn \l__enumext_store_columns_join_int {#1}
2035            \bool_if:NT \l__enumext_starred_bool
2036              {
2037                \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2038                  {
2039                    ( \exp_not:V \l__enumext_store_columns_join_int )
2040                  }
2041              }
2042          }
```

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`.

```
2043        \bool_if:nTF {#2}
2044          {
2045            \tl_put_right:Nn \l__enumext_store_anskey_arg_tl { * }
2046            \tl_if_empty:NF \l__enumext_store_item_symbol_tl
2047              {
2048                \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2049                  {
2050                    [ \exp_not:V \l__enumext_store_item_symbol_tl ]
2051                  }
2052              }
2053            \dim_compare:nT
2054              {
2055                \l__enumext_store_item_symbol_sep_dim != \c_zero_dim
2056              }
2057              {
2058                \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2059                  {
2060                    [ \exp_not:V \l__enumext_store_item_symbol_sep_dim ]
2061                  }
2062              }
2063            \tl_put_right:Nn \l__enumext_store_anskey_arg_tl {#4}
2064          }
2065          {
2066            \tl_put_right:Nn \l__enumext_store_anskey_arg_tl {#4}
2067          }
```

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*⟩.

```
2068        \bool_lazy_and:nnT
2069          { \bool_if_p:N \l__enumext_store_ref_key_bool }
2070          { \bool_if_p:N \l__enumext_hyperref_bool }
2071          {
2072            \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2073              {
2074                \hfill \exp_not:N \hyperlink { \exp_not:V \l__enumext_newlabel_arg_one_tl }
2075                  { \exp_not:V \l__enumext_mark_ref_sym_tl }
2076              }
2077          }
2078        \__enumext_store_addto_seq:V \l__enumext_store_anskey_arg_tl
2079      }
```

(*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.

```
2080  \cs_new_protected:Nn \__enumext_store_internal_ref:
2081    {
2082      \cs_set_protected:Npn \__enumext_tmp:n ##1
2083        {
2084          \tl_set_eq:cc { l__enumext_label_copy_##1_tl } { l__enumext_label_##1_tl }
2085          \tl_reverse:c { l__enumext_label_copy_##1_tl }
2086          \tl_remove_once:cn { l__enumext_label_copy_##1_tl } { . }
2087          \tl_reverse:c { l__enumext_label_copy_##1_tl }
2088        }
2089      \clist_map_inline:nn { i, ii, iii, iv, vii } { \__enumext_tmp:n {##1} }
2090      \cs_set:Npn \__enumext_tmp:n ##1
2091        { . \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.

```
2092      \bool_lazy_all:nT
2093        {
2094          { \bool_if_p:N \g__enumext_starred_bool }
2095          { \int_compare_p:nNn { \l__enumext_level_int } = { \c_zero_int } }
2096        }
2097        {
2098          \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2099            { \tl_use:N \l__enumext_label_copy_vii_tl }
2100        }
2101      \bool_lazy_all:nT
2102        {
2103          { \bool_if_p:N \l__enumext_standar_bool }
2104          { \bool_if_p:N \g__enumext_starred_bool }
2105          { \int_compare_p:nNn { \l__enumext_level_int } > { \c_zero_int } }
2106        }
2107        {
2108          \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2109            {
2110              \tl_use:N \l__enumext_label_copy_vii_tl
2111              \int_step_function:nnN { 1 } { \l__enumext_level_int } \__enumext_tmp:n
2112            }
2113        }
```

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.

```
2114      \bool_lazy_all:nT
2115        {
2116          { \bool_if_p:N \l__enumext_standar_bool }
2117          { \int_compare_p:nNn { \l__enumext_level_int } > { \c_zero_int } }
2118          { \int_compare_p:nNn { \l__enumext_level_h_int } = { \c_zero_int } }
2119          { \bool_not_p:n { \l__enumext_starred_bool } }
2120        }
2121        {
2122          \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2123            {
2124              \tl_use:N \l__enumext_label_copy_i_tl
2125              \int_step_function:nnN { 2 } { \l__enumext_level_int } \__enumext_tmp:n
2126            }
2127        }
2128      \cs_set:Npn \__enumext_tmp:n ##1
2129        { \tl_use:c { l__enumext_label_copy_ \int_to_roman:n {##1} _tl } }
2130      \bool_lazy_all:nT
2131        {
2132          { \bool_if_p:N \l__enumext_standar_bool }
2133          { \int_compare_p:nNn { \l__enumext_level_int } > { \c_zero_int } }
2134          { \bool_not_p:n { \g__enumext_starred_bool } }
2135          { \int_compare_p:nNn { \l__enumext_level_h_int } > { \c_zero_int } }
2136        }
2137        {
2138          \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2139            {
```

```
2140            \int_step_function:nnN { 1 } { \l__enumext_level_int } \__enumext_tmp:n
2141              . \tl_use:N \l__enumext_label_copy_vii_tl
2142          }
2143        }
```

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

```
2144      \tl_put_right:Ne \l__enumext_newlabel_arg_one_tl
2145        {
2146          \l__enumext_store_name_tl \c_colon_str
2147          \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } }
2148        }
```

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.

```
2149      \tl_put_right:Ne \l__enumext_store_write_aux_file_tl
2150        {
2151          \__enumext_newlabel:nn
2152            { \exp_not:V \l__enumext_newlabel_arg_one_tl }
2153            { \l__enumext_newlabel_arg_two_tl }
2154        }
2155      \l__enumext_store_write_aux_file_tl
2156    }
```

(*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.

```
2157  \cs_new_protected:Npn \__enumext_store_anskey_show_wrap:n #1
2158    {
2159      \par
2160      \bool_if:NT \l__enumext_starred_bool
2161        {
2162          \cs_set:Nn \__enumext_level: { vii }
2163        }
2164      \__enumext_print_keyans_box:cc
2165        { l__enumext_labelwidth_ \__enumext_level: _dim }
2166        { l__enumext_labelsep_ \__enumext_level: _dim }
2167      \__enumext_anskey_wrapper:n { #1 }
2168    }
```

(*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.

```
2169  \cs_new_protected:Npn \__enumext_store_anskey_show_left:n #1
2170    {
2171      \bool_if:NT \l__enumext_show_answer_bool
2172        {
2173          \__enumext_store_anskey_show_wrap:n { #1 }
2174        }
2175      \bool_if:NT \l__enumext_show_position_bool
2176        {
2177          \tl_set:Ne \l__enumext_mark_answer_sym_tl
2178            {
2179              \group_begin:
2180              \exp_not:N \normalfont
2181              \exp_not:N \footnotesize [ \int_eval:n
2182                {
2183                  \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
2184                }
2185                ]
2186              \group_end:
2187            }
2188          \__enumext_store_anskey_show_wrap:n { #1 }
2189        }
2190    }
```

(*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`.

```
2191  \cs_new_protected:Npn \__enumext_keyans_addto_prop:n #1
2192    {
2193      \tl_clear:N \l__enumext_store_keyans_label_tl
2194      \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
2195        {
2196          \tl_put_right:Ne \l__enumext_store_keyans_label_tl { \l__enumext_label_vi_tl }
2197        }
2198        {
2199          \tl_put_right:Ne \l__enumext_store_keyans_label_tl { \l__enumext_label_v_tl }
2200        }
2201      \tl_if_novalue:nF { #1 }
2202        {
2203          % Set save-sep
2204          \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_tl
2205            {
2206              \tl_put_right:Ne \l__enumext_store_keyans_label_tl { \l__enumext_store_keyans_item_op
2207            }
2208          \tl_put_right:Ne \l__enumext_store_keyans_label_tl { #1 }
2209        }
2210      \__enumext_store_addto_prop:V \l__enumext_store_keyans_label_tl
2211    }
```

(*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.

```
2212  \cs_new_protected:Nn \__enumext_keyans_store_ref:
2213    {
2214      \bool_if:NT \l__enumext_store_ref_key_bool
2215        {
2216          \cs_set_protected:Npn \__enumext_tmp:n ##1
2217            {
2218              \tl_set_eq:cc { l__enumext_label_copy_##1_tl } { l__enumext_label_##1_tl }
2219              \tl_reverse:c { l__enumext_label_copy_##1_tl }
2220              \tl_remove_once:cn { l__enumext_label_copy_##1_tl } { . }
2221              \tl_reverse:c { l__enumext_label_copy_##1_tl }
2222            }
2223          \clist_map_inline:nn { i, v, vi, vii, viii } { \__enumext_tmp:n {##1} }
2224          \__enumext_keyans_store_ref_aux_i:
2225        }
2226    }
```

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`.

```
2227  \cs_new_protected:Nn \__enumext_keyans_store_ref_aux_i:
2228    {
2229      \bool_if:NT \g__enumext_starred_bool
2230        {
2231          \tl_set_eq:NN \l__enumext_label_copy_i_tl \l__enumext_label_copy_vii_tl
2232        }
2233      \int_compare:nNnT { \l__enumext_keyans_pic_level_int } = { 1 }
2234        {
2235          \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
```

```
2236                { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_vi_tl }
2237              }
2238          \int_compare:nNnT { \l__enumext_keyans_level_int } = { 1 }
2239            {
2240              \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2241                { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_v_tl }
2242            }
2243          \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
2244            {
2245              \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2246                { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_viii_tl }
2247            }
2248          \tl_put_right:Ne \l__enumext_newlabel_arg_one_tl
2249            {
2250              \l__enumext_store_name_tl \c_colon_str
2251              \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } }
2252            }
2253          \__enumext_keyans_store_ref_aux_ii:
2254      }
```

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.

```
2255  \cs_new_protected:Nn \__enumext_keyans_store_ref_aux_ii:
2256      {
2257        \tl_put_right:Ne \l__enumext_store_write_aux_file_tl
2258          {
2259            \__enumext_newlabel:nn
2260              { \exp_not:V \l__enumext_newlabel_arg_one_tl }
2261              { \l__enumext_newlabel_arg_two_tl }
2262          }
2263        \l__enumext_store_write_aux_file_tl
2264      }
```

(*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.

```
2265  \cs_new_protected:Npn \__enumext_keyans_addto_seq:n #1
2266      {
2267        \tl_clear:N \l__enumext_store_keyans_label_tl
2268        \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
2269          {
2270            \tl_put_right:Ne \l__enumext_store_keyans_label_tl { \item \l__enumext_label_vi_tl }
2271          }
2272          {
2273            \tl_put_right:Ne \l__enumext_store_keyans_label_tl { \item \l__enumext_label_v_tl }
2274          }
2275        \tl_if_novalue:nF { #1 }
2276          {
2277            \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_tl
2278              {
2279                \tl_put_right:Ne \l__enumext_store_keyans_label_tl { \l__enumext_store_keyans_item_opt
2280              }
2281            \tl_put_right:Ne \l__enumext_store_keyans_label_tl { #1 }
2282          }
2283        \__enumext_keyans_addto_seq_link:
2284      }
```

Checks if the save-ref key is active along with the hyperref package load, if both conditions are met, it will create the `\hyperlink` and then store using the `\__enumext_store_addto_seq:V` function. Finally, copy the contents of the variable `\l__enumext_store_keyans_label_tl` into the global variable `\g__enumext_check_ans_item_tl` to be used by the function `\__enumext_check_starred_cmd:nn` and increment the value of the integer variable `\g__enumext_count_item_anskey_int` handled by the check-ans key.

```
2285  \cs_new_protected:Nn \__enumext_keyans_addto_seq_link:
2286      {
```

```
2287    \bool_lazy_and:nnT
2288      { \bool_if_p:N \l__enumext_store_ref_key_bool }
2289      { \bool_if_p:N \l__enumext_hyperref_bool }
2290      {
2291        \tl_put_right:Ne \l__enumext_store_keyans_label_tl
2292          {
2293            \hfill \exp_not:N \hyperlink
2294              {
2295                \exp_not:V \l__enumext_newlabel_arg_one_tl
2296              }
2297              { \exp_not:V \l__enumext_mark_ref_sym_tl }
2298          }
2299      }
2300    \__enumext_store_addto_seq:V \l__enumext_store_keyans_label_tl
2301    \tl_gset:NV \g__enumext_check_ans_item_tl \l__enumext_store_keyans_label_tl
2302    \bool_if:NT \l__enumext_check_ans_bool
2303      {
2304        \int_gincr:N \g__enumext_count_item_anskey_int
2305      }
2306  }
```

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

### 10.26.4  The show-ans and show-pos keys for keyans and keyanspic

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

\__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_ ℓ__enumext_store_name_tl _prop for show-pos key.

```
2307  \cs_new_protected:Npn \__enumext_keyans_show_left:n #1
2308    {
2309      \tl_if_novalue:nF { #1 }
2310        {
2311          \tl_set:Ne \l__enumext_keyans_item_opt_tl { #1 }
2312        }
2313      \bool_if:NT \l__enumext_show_answer_bool
2314        {
2315          \__enumext_keyans_show_ans:
2316        }
2317      \bool_if:NT \l__enumext_show_position_bool
2318        {
2319          \__enumext_keyans_show_pos:
2320        }
2321    }
2322  \cs_new_protected:Nn \__enumext_keyans_show_item_opt:
2323    {
2324      \tl_if_empty:NF \l__enumext_keyans_item_opt_tl
2325        {
2326          \bool_lazy_or:nnT
2327            { \bool_if_p:N \l__enumext_show_answer_bool }
2328            { \bool_if_p:N \l__enumext_show_position_bool }
2329            {
2330              \__enumext_keyans_wrapper_opt:n { \l__enumext_keyans_item_opt_tl } \c_space_tl
2331            }
2332        }
2333    }
2334  \cs_new_protected:Nn \__enumext_keyans_show_ans:
2335    {
2336      \tl_put_left:Nn \l__enumext_label_v_tl
2337        {
2338          \__enumext_print_keyans_box:NN \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
2339        }
2340    }
2341  \cs_new_protected:Nn \__enumext_keyans_show_pos:
2342    {
2343      \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
2344        {
2345          \tl_set:Ne \l__enumext_mark_answer_sym_tl
2346            {
2347              \group_begin:
```

```
2348              \exp_not:N \normalfont
2349              \exp_not:N \footnotesize [ \int_eval:n
2350                {
2351                  \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
2352                }
2353                ]
2354              \group_end:
2355            }
2356        }
2357        {
2358          \tl_set:Ne \l__enumext_mark_answer_sym_tl
2359            {
2360              \group_begin:
2361              \exp_not:N \normalfont
2362              \exp_not:N \footnotesize [ \int_eval:n
2363                {
2364                  \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } + 1
2365                }
2366                ]
2367              \group_end:
2368            }
2369        }
2370    \tl_put_left:Nn \l__enumext_label_v_tl
2371      {
2372        \__enumext_print_keyans_box:NN
2373          \l__enumext_labelwidth_i_dim
2374          \l__enumext_labelsep_i_dim
2375      }
2376  }
```

(*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*.

```
2377 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
2378   {
2379     \keys_define:nn { enumext / #1 }
2380       {
2381         item-sym* .tl_set:c  = { l__enumext_item_symbol_#2_tl },
2382         item-sym* .value_required:n = true,
2383         item-sym* .initial:n  = {$\star$},
2384         item-pos* .dim_set:c  = { l__enumext_item_symbol_sep_#2_dim },
2385         item-pos* .value_required:n = true,
2386       }
2387   }
2388 \clist_map_inline:nn
2389   {
2390     {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {enumext*}{vii}
2391   }
2392   { \__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.

```
2393 \cs_new_protected:Nn \__enumext_footnotetext:nn
2394   {
2395     \footnotetext[#1]{#2}
2396   }
2397 \cs_new_protected:Nn \__enumext_renew_footnote:
2398   {
2399     \seq_gclear:N \g__enumext_footnote_arg_seq
2400     \seq_gclear:N \g__enumext_footnote_int_seq
2401     \RenewDocumentCommand \footnote { o +m }
2402       {
```

```
2403        \tl_if_novalue:nTF {##1}
2404          {
2405            \stepcounter{footnote}
2406            \int_gset_eq:Nc \g__enumext_footnote_int { c@footnote }
2407          }
2408          {
2409            \int_gset:Nn \g__enumext_footnote_int { ##1 }
2410          }
2411        \footnotemark [ \g__enumext_footnote_int ]
2412        \seq_gput_right:Nn \g__enumext_footnote_arg_seq { ##2 }
2413        \seq_gput_right:NV \g__enumext_footnote_int_seq \g__enumext_footnote_int
2414      }
2415    }
2416 \cs_new_protected:Nn \__enumext_print_footnote:
2417    {
2418      \seq_if_empty:NF \g__enumext_footnote_int_seq
2419        {
2420          \seq_map_pairwise_function:NNN
2421            \g__enumext_footnote_int_seq
2422            \g__enumext_footnote_arg_seq
2423            \__enumext_footnotetext:nn
2424        }
2425    }
```

(*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).

```
2426 \cs_new_protected:Npn \__enumext_default_item:n #1
2427    {
2428      \tl_if_novalue:nTF {#1}
2429        {
2430          \bool_if:NT \l__enumext_check_ans_bool
2431            {
2432              \int_gincr:N \g__enumext_count_item_number_int
2433            }
2434          \bool_set_true:c { l__enumext_wrap_label_ \__enumext_level: _bool }
2435          \__enumext_item_std:w \tl_use:c { l__enumext_fake_item_indent_ \__enumext_level: _tl }
2436        }
2437        {
2438          \bool_set_eq:cc
2439            { l__enumext_wrap_label_ \__enumext_level: _bool }
2440            { l__enumext_wrap_label_opt_ \__enumext_level: _bool }
2441          \__enumext_item_std:w [#1] \tl_use:c { l__enumext_fake_item_indent_ \__enumext_level: _tl
2442        }
2443    }
```

(*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).

```
2444  \cs_new_protected:Npn \__enumext_starred_item:nn #1 #2
2445    {
2446      \tl_if_novalue:nF {#1}
2447        {
2448          \tl_set:cn { l__enumext_item_symbol_ \__enumext_level: _tl } {#1}
2449        }
2450      \tl_gset_eq:Nc \g__enumext_item_symbol_tl { l__enumext_item_symbol_ \__enumext_level: _tl }
2451      \tl_if_novalue:nTF {#2}
2452        {
2453          \dim_set_eq:cc
2454            { l__enumext_item_symbol_sep_ \__enumext_level: _dim }
2455            { l__enumext_labelsep_ \__enumext_level: _dim }
2456        }
2457        {
2458          \dim_set:cn { l__enumext_item_symbol_sep_ \__enumext_level: _dim } {#2}
2459        }
2460      \bool_if:NT \l__enumext_check_ans_bool
2461        {
2462          \int_gincr:N \g__enumext_count_item_number_int
2463        }
2464      \bool_set_true:c { l__enumext_wrap_label_ \__enumext_level: _bool }
2465      \__enumext_item_std:w \tl_use:c { l__enumext_fake_item_indent_ \__enumext_level: _tl }
2466    }
```

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

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

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

```
2467  \cs_new_protected:Nn \__enumext_redefine_item:
2468    {
2469      \RenewDocumentCommand \item { s o o }
2470        {
2471          \bool_if:nTF {##1}
2472            {
2473              \__enumext_starred_item:nn {##2} {##3}
2474            }
2475            { \__enumext_default_item:n {##2} }
2476        }
2477    }
```

(*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`.

```
2478  \cs_new_protected:Npn \__enumext_keyans_default_item:n #1
2479    {
2480      \tl_if_novalue:nTF { #1 }
2481        {
2482          \bool_set_true:N \l__enumext_wrap_label_v_bool
2483          \__enumext_item_std:w \tl_use:N \l__enumext_fake_item_indent_v_tl
```

```
2484          }
2485          {
2486              \bool_set_eq:NN \l__enumext_wrap_label_v_bool \l__enumext_wrap_label_opt_v_bool
2487              \__enumext_item_std:w [#1] \tl_use:N \l__enumext_fake_item_indent_v_tl
2488          }
2489      }
```

(*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*⟩.

```
2490 \cs_new_protected:Npn \__enumext_keyans_starred_item:n #1
2491    {
2492      \tl_set_eq:NN \l__enumext_keyans_tmpa_tl \l__enumext_label_v_tl
2493      \__enumext_keyans_show_left:n { #1 }
2494      \bool_set_true:N \l__enumext_wrap_label_v_bool
2495      \__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*⟩.

```
2496      \tl_set_eq:NN \l__enumext_label_v_tl \l__enumext_keyans_tmpa_tl
2497      \__enumext_keyans_addto_prop:n { #1 }
2498      \__enumext_keyans_store_ref:
2499      \__enumext_keyans_addto_seq:n { #1 }
2500      \int_incr:N \l__enumext_check_starred_cmd_int
2501    }
```

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

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

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

```
2502 \cs_new_protected:Nn \__enumext_keyans_redefine_item:
2503    {
2504      \RenewDocumentCommand \item { s o }
2505        {
2506          \bool_if:nTF {##1}
2507            {
2508              \peek_remove_spaces:n
2509                {
2510                  \__enumext_keyans_starred_item:n {##2}
2511                }
2512            }
2513            {
2514              \__enumext_keyans_default_item:n {##2}
2515            }
2516        }
2517    }
```

(*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.

```
2518 \cs_new_protected:Nn \__enumext_item_starred:
2519    {
2520      \tl_if_empty:cF { l__enumext_item_symbol_ \__enumext_level: _tl }
2521        {
2522          \mode_leave_vertical:
2523          \skip_horizontal:n { -\dim_use:c { l__enumext_item_symbol_sep_ \__enumext_level: _dim } }
```

```
2524            \makebox[ 0pt ][ r ]{ \g__enumext_item_symbol_tl }
2525            \skip_horizontal:n { \dim_use:c { l__enumext_item_symbol_sep_ \__enumext_level: _dim } }
2526          }
2527        }
```

(*End of definition for* \__enumext_item_starred:.)

\__enumext_make_label:

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

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

```
2528 \cs_new_protected:Nn \__enumext_make_label:
2529   {
2530     \RenewDocumentCommand \makelabel { m }
2531       {
2532         \tl_use:c { l__enumext_label_fill_left_ \__enumext_level: _tl }
2533         \tl_use:c { l__enumext_label_font_style_ \__enumext_level: _tl }
2534         \bool_if:cTF { l__enumext_wrap_label_ \__enumext_level: _bool }
2535           {
2536             \__enumext_item_starred:
2537             \use:c { __enumext_wrapper_label_ \__enumext_level: :n } { ##1 }
2538           }
2539           { ##1 }
2540         \tl_use:c { l__enumext_label_fill_right_ \__enumext_level: _tl }
2541         \tl_gclear:N \g__enumext_item_symbol_tl
2542       }
2543   }
```

(*End of definition for* \__enumext_make_label:.)

### 10.30.2   Redefining \makelabel **for keyans**

\__enumext_keyans_make_label:

The function \__enumext_keyans_make_label: redefine \makelabel for keyans environment.

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

```
2544 \cs_new_protected:Nn \__enumext_keyans_make_label:
2545   {
2546     \RenewDocumentCommand \makelabel { m }
2547       {
2548         \tl_use:N \l__enumext_label_fill_left_v_tl
2549         \tl_use:N \l__enumext_label_font_style_v_tl
2550         \bool_if:NTF \l__enumext_wrap_label_v_bool
2551           {
2552             \__enumext_wrapper_label_v:n { ##1 }
2553           }
2554           { ##1 }
2555         \tl_use:N \l__enumext_label_fill_right_v_tl
2556       }
2557   }
```

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

## 10.31   Second argument of the lists

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

### 10.31.1   Calculation of \leftmargin **and** \itemindent

Consider the figure 9 where the default margins (on the left) of a list are represented.
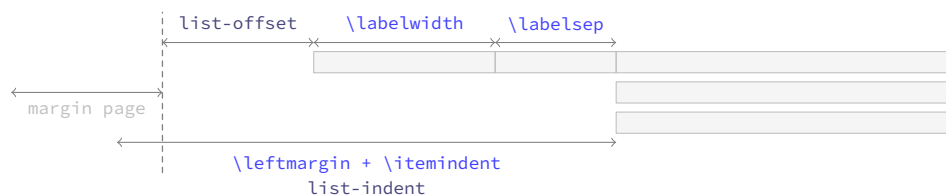


Figure 9: Representation of standard horizontal lengths in list environment.

The idea is to have control over these margins so that our list does not overlap the left margin of the page. The *key* relationship is that the right edge of the \labelsep equals the right edge of the \itemindent,
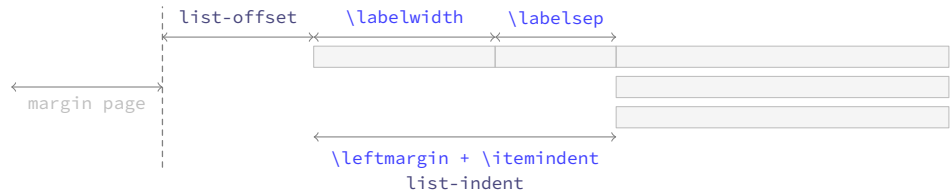
Figure 10: Representation of horizontal lengths concept in list in enumext.

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.
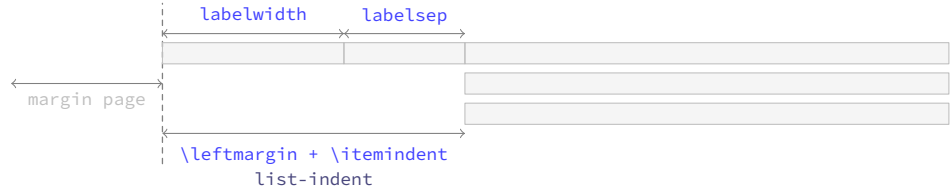


Figure 11: Default horizontal lengths in enumext.

`\__enumext_calc_hspace:NNNNNNN`
`\__enumext_calc_hspace:ccccccc`

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

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

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

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

```
2558 \cs_new_protected:Npn \__enumext_calc_hspace:NNNNNNN #1 #2 #3 #4 #5 #6 #7
2559   {
2560     \dim_compare:nNnT { #1 } < { \c_zero_dim }
2561       {
2562         \msg_warning:nnnV { enumext } { width-non-positive }{ labelwidth }{ #1 }
2563         \dim_set:Nn #1 { \dim_abs:n { #1 } }
2564       }
2565     \dim_compare:nNnT { #2 } < { \c_zero_dim }
2566       {
2567         \msg_warning:nnnV { enumext } { width-negative }{ labelsep }{ #2 }
2568         \dim_set:Nn #2 { \dim_abs:n { #2 } }
2569       }
```

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

```
2570     \bool_if:nF #7 { \dim_set:Nn #4 { #1 + #2 } }
```

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

```
2571     \dim_compare:nNnTF { #4 } < { \c_zero_dim }
2572       {
2573         \dim_set:Nn #6 { #1 + #2 - #4}
2574         \dim_set:Nn #5 { #1 + #2 + #3 - #6 }
2575       }
2576       {
2577         \dim_compare:nNnT { #4 } = { #1 + #2 }
2578           { \dim_set:Nn #6 { \c_zero_dim } }
2579         \dim_compare:nNnT { #4 } < { #1 + #2 }
2580           { \dim_set:Nn #6 { #1 + #2 - #4} }
2581         \dim_compare:nNnT { #4 } > { #1 + #2 }
2582           {
2583             \dim_set:Nn #6 { -#1 - #2 + #4}
2584             \dim_set:Nn #6 { #6*-1}
2585           }
2586         \dim_set:Nn #5 { #1 + #2 + #3 - #6 }
2587       }
2588   }
2589 \cs_generate_variant:Nn \__enumext_calc_hspace:NNNNNNN { ccccccc }
```

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

### 10.31.2 Setting second argument of the lists

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

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

```
2590 \cs_set_protected:Npn \__enumext_tmp:n #1
2591   {
2592     \cs_new_protected:cpn { __enumext_list_arg_two_#1: }
2593       {
2594         \__enumext_calc_hspace:ccccccc
2595           { l__enumext_labelwidth_#1_dim } { l__enumext_labelsep_#1_dim }
2596           { l__enumext_listoffset_#1_dim } { l__enumext_leftmargin_tmp_#1_dim }
2597           { l__enumext_leftmargin_#1_dim } { l__enumext_itemindent_#1_dim }
2598           { l__enumext_leftmargin_tmp_#1_bool }
2599         \clist_map_inline:nn
2600           { labelsep, labelwidth, itemindent, leftmargin, rightmargin, listparindent }
2601           { \dim_set_eq:cc {####1} { l__enumext_####1_#1_dim } }
2602         \clist_map_inline:nn { topsep, parsep, partopsep, itemsep }
2603           { \skip_set_eq:cc {####1} { l__enumext_####1_#1_skip } }
2604         \usecounter { enumX#1 }
2605         \setcounter { enumX#1 } { \int_eval:n { \int_use:c { l__enumext_start_#1_int } - 1 } }
2606         \str_if_eq:nnTF {#1} { v }
2607           {
2608             \__enumext_keyans_redefine_item:
2609             \__enumext_keyans_make_label:
2610             \__enumext_keyans_ref:
2611             \__enumext_keyans_fake_item:
2612             \bool_if:cT { l__enumext_show_length_#1_bool }
2613               {
2614                 \msg_term:nnnn { enumext } { list-lengths-not-nested } { v } { keyans }
2615               }
2616           }
2617           {
2618             \__enumext_redefine_item:
2619             \__enumext_make_label:
2620             \__enumext_standar_ref:
2621             \__enumext_fake_item:
2622             \bool_if:cT { l__enumext_show_length_#1_bool }
2623               {
2624                 \msg_term:nnne { enumext } { list-lengths } {#1} { \int_use:N \l__enumext_level_i
2625               }
2626           }
2627       }
2628   }
2629 \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.

```
2630 \cs_set_protected:Npn \__enumext_tmp:n #1
2631   {
2632     \cs_new_protected:cpn { __enumext_list_arg_two_#1: }
2633       {
2634         \__enumext_calc_hspace:ccccccc
2635           { l__enumext_labelwidth_#1_dim } { l__enumext_labelsep_#1_dim }
2636           { l__enumext_listoffset_#1_dim } { l__enumext_leftmargin_tmp_#1_dim }
2637           { l__enumext_leftmargin_#1_dim } { l__enumext_itemindent_#1_dim }
2638           { l__enumext_leftmargin_tmp_#1_bool }
2639         \clist_map_inline:nn
2640           { labelsep, labelwidth, itemindent, leftmargin, rightmargin, listparindent }
2641           { \dim_set_eq:cc {####1} { l__enumext_####1_#1_dim } }
2642         \clist_map_inline:nn { topsep, parsep, partopsep, itemsep }
2643           { \skip_set_eq:cc {####1} { l__enumext_####1_#1_skip } }
2644         \skip_set_eq:Nc \parsep  { l__enumext_itemsep_#1_skip }
2645         \skip_zero:N \partopsep
2646         \usecounter { enumX#1 }
2647         \setcounter { enumX#1 } { \int_eval:n { \int_use:c { l__enumext_start_#1_int } - 1 } }
2648         \__enumext_starred_ref:
2649         \str_if_eq:nnTF {#1} { vii }
```

```
2650                {
2651                    \__enumext_fake_item_vii:
2652                    \bool_if:cT { l__enumext_show_length_vii_bool }
2653                        { \msg_term:nnnn { enumext } { list-lengths-not-nested } { vii } { enumext* } }
2654                }
2655                {
2656                    \__enumext_fake_item_viii:
2657                    \bool_if:cT { l__enumext_show_length_#1_bool }
2658                        { \msg_term:nnnn { enumext } { list-lengths-not-nested } { #1 } { keyans* } }
2659                }
2660            }
2661        }
2662    \clist_map_inline:nn { vii, viii } { \__enumext_tmp:n {#1} }
```

(*End of definition for* \__enumext_list_arg_two_vii: *and* \__enumext_list_arg_two_viii:.)

### 10.32 The environment enumext

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

```
2663    \NewDocumentEnvironment{enumext}{ O{} }
2664        {
2665            \__enumext_safe_exec:
2666            \__enumext_parse_keys:n {#1}
2667            \__enumext_before_list:
2668            \__enumext_start_store_level:
2669            \__enumext_start_list:nn
2670                { \tl_use:c { l__enumext_label_ \__enumext_level: _tl } }
2671                {
2672                    \use:c { __enumext_list_arg_two_ \__enumext_level: : }
2673                    \__enumext_before_keys_exec:
2674                }
2675            \__enumext_after_args_exec:
2676        }
2677        {
2678            \__enumext_stop_list:
2679            \__enumext_stop_store_level:
2680            \__enumext_after_list:
2681        }
```

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

\__enumext_safe_exec: The \__enumext_safe_exec: function first execute the function \__enumext_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 using the function \__enumext_check_first_level:.

```
2682    \cs_new_protected:Nn \__enumext_safe_exec:
2683        {
2684            \__enumext_current_env_set_bool:
2685            \int_incr:N \l__enumext_level_int
2686            \int_compare:nNnT { \l__enumext_level_int } > { 4 }
2687                { \msg_fatal:nn { enumext } { list-too-deep } }
2688            \bool_set_true:N \l__enumext_standar_bool
2689            \__enumext_check_first_level:
2690        }
```

(*End of definition for* \__enumext_safe_exec:.)

\__enumext_parse_keys:n The \__enumext_parse_store_keys:n function will parse the ⟨*keys*⟩ passed to the optional environment argument enumext by levels only if present. First we clear the variable \l__enumext_series_str and then we check if we are at the first level, if so we process the ⟨*keys*⟩ and then execute the function \__enumext_parse_series:n used by the key series, otherwise we will pass the ⟨*keys*⟩ to the inner levels of the environment and finally if the variable \l__enumext_store_active_bool established by the key save-ans is true we execute \__enumext_parse_store_keys:n used by the key save-key.

```
2691    \cs_new_protected:Npn \__enumext_parse_keys:n #1
2692        {
2693            \tl_if_novalue:nF {#1}
2694                {
2695                    \str_clear:N \l__enumext_series_str
2696                    \int_compare:nNnTF { \l__enumext_level_int } = { 1 }
```

```
2697            {
2698              \keys_set:nn { enumext / level-1 } {#1}
2699              \__enumext_parse_series:n {#1}
2700            }
2701            {
2702              \exp_args:Ne \keys_set:nn
2703                { enumext / level-\int_use:N \l__enumext_level_int } {#1}
2704            }
2705          \bool_if:NT \l__enumext_store_active_bool
2706            {
2707              \__enumext_parse_store_keys:n {#1}
2708            }
2709        }
2710    }
```

(*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:.

```
2711  \cs_new_protected:Npn \__enumext_parse_store_keys:n #1
2712    {
2713      \bool_if:cF { l__enumext_store_columns_ \__enumext_level: _bool }
2714        {
2715          \regex_match:nnT { \b columns\b } {#1}
2716            {
2717              \int_set_eq:cc
2718                { l__enumext_store_columns_ \__enumext_level: _int }
2719                { l__enumext_columns_  \__enumext_level: _int }
2720              \tl_put_right:ce { l__enumext_store_opt_ \__enumext_level: _tl }
2721                {
2722                  columns = \exp_not:v { l__enumext_store_columns_ \__enumext_level: _int },
2723                }
2724            }
2725        }
2726      \bool_if:cF { l__enumext_store_columns_sep_ \__enumext_level: _bool }
2727        {
2728          \regex_match:nnT { \b columns-sep \b} {#1}
2729            {
2730              \dim_set_eq:cc
2731                { l__enumext_store_columns_sep_ \__enumext_level: _dim }
2732                { l__enumext_columns_sep_  \__enumext_level: _dim }
2733              \tl_put_right:ce { l__enumext_store_opt_ \__enumext_level: _tl }
2734                {
2735                  columns-sep = \exp_not:v { l__enumext_store_columns_sep_ \__enumext_level: _dim }
2736                }
2737            }
2738        }
2739    }
```

(*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.

```
2740  \cs_new_protected:Nn \__enumext_start_store_level:
2741    {
2742      \bool_lazy_all:nT
2743        {
2744          { \bool_if_p:N \l__enumext_store_active_bool }
2745          { \bool_not_p:n { \l__enumext_keyans_env_bool } }
2746          { \bool_not_p:n { \g__enumext_starred_bool } }
2747        }
2748        {
2749          \int_compare:nNnT { \l__enumext_level_int } > { 1 }
2750            {
2751              \bool_set_true:c { l__enumext_store_upper_level_ \__enumext_level: _bool }
2752              \__enumext_store_level_open:
2753            }
```

```
2754              }
2755          \bool_lazy_all:nT
2756            {
2757              { \bool_if_p:N \l__enumext_store_active_bool }
2758              { \bool_not_p:n { \l__enumext_keyans_env_bool } }
2759              { \bool_if_p:N \g__enumext_starred_bool }
2760            }
2761            {
2762              \int_compare:nNnT { \l__enumext_level_int } > { 0 }
2763                {
2764                  \bool_set_true:c { l__enumext_store_upper_level_ \__enumext_level: _bool }
2765                  \__enumext_store_level_open:
2766                }
2767            }
2768        }
2769    \cs_new_protected:Nn \__enumext_stop_store_level:
2770      {
2771        \bool_if:cT { l__enumext_store_upper_level_ \__enumext_level: _bool }
2772          {
2773            \__enumext_store_level_close:
2774          }
2775      }
```

*(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.

```
2776    \cs_new_protected:Nn \__enumext_before_list:
2777      {
2778        \__enumext_vspace_above:
2779        \__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.

```
2780        \__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.

```
2781        \dim_compare:nNnT
2782          { \dim_use:c { l__enumext_minipage_right_ \__enumext_level: _dim } } > { \c_zero_dim }
2783          {
2784            \dim_set:cn { l__enumext_minipage_left_ \__enumext_level: _dim }
2785              {
2786                \linewidth
2787                - \dim_use:c { l__enumext_minipage_right_ \__enumext_level: _dim }
2788                - \dim_use:c { l__enumext_minipage_hsep_ \__enumext_level: _dim }
2789              }
```

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*.

```
2790            \bool_set_true:c { l__enumext_minipage_active_ \__enumext_level: _bool }
2791            \int_gincr:N \g__enumext_minipage_stat_int
2792            \__enumext_mini_addvspace:
2793            \nointerlineskip\noindent
2794            \begin{__enumext_mini_env*}
2795              { \dim_use:c { l__enumext_minipage_left_ \__enumext_level: _dim } }
2796          }
2797        \__enumext_multicols_start:
2798      }
```

*(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.

```
2799 \cs_new_protected:Nn \__enumext_multicols_start:
2800   {
2801     \int_compare:nNnT
2802       { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } > { 1 }
2803       {
2804         \dim_compare:nNnT
2805           { \dim_use:c { l__enumext_columns_sep_ \__enumext_level: _dim } } = { \c_zero_dim }
2806           {
2807             \dim_set:cn { l__enumext_columns_sep_ \__enumext_level: _dim }
2808               {
2809                 ( \dim_use:c { l__enumext_labelwidth_ \__enumext_level: _dim }
2810                   + \dim_use:c { l__enumext_labelsep_ \__enumext_level: _dim }
2811                 ) / \int_use:c { l__enumext_columns_ \__enumext_level: _int }
2812                 - \dim_use:c { l__enumext_listoffset_ \__enumext_level: _dim }
2813               }
2814           }
2815         \dim_set_eq:Nc \columnsep { l__enumext_columns_sep_ \__enumext_level: _dim  }
2816         \skip_zero:N \multicolsep
2817         \int_compare:nNnT { \l__enumext_level_int } > { 1 }
2818           {
2819             \dim_zero:N \columnseprule
2820           }
```

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.

```
2821         \bool_if:cF { l__enumext_minipage_active_ \__enumext_level: _bool }
2822           {
2823             \__enumext_multi_addvspace:
2824           }
2825         \raggedcolumns
2826         \begin{multicols}{ \int_use:c { l__enumext_columns_ \__enumext_level: _int } }
2827       }
2828   }
```

(*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.

```
2829 \cs_new_protected:Nn \__enumext_multicols_stop:
2830   {
2831     \int_compare:nNnT
2832       { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } > { 1 }
2833       {
2834         \end{multicols}
2835         \bool_if:cF { l__enumext_minipage_active_ \__enumext_level: _bool }
2836           {
2837             \par\addvspace{ \skip_use:c { l__enumext_multicols_below_ \__enumext_level: _skip } }
2838           }
2839       }
2840   }
```

(*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.

```
2841 \cs_new_protected:Nn \__enumext_after_list:
2842   {
2843     \bool_if:cTF { l__enumext_minipage_active_ \__enumext_level: _bool }
2844       {
2845         \int_compare:nNnT { \g__enumext_minipage_stat_int } = { 1 }
2846           {
```

```
2847              \msg_warning:nn { enumext } { missing-miniright }
2848              \miniright
2849            }
2850          \int_gzero:N \g__enumext_minipage_stat_int
2851          \end{__enumext_mini_env*}
2852          \par\addvspace { \l__enumext_minipage_after_skip }
2853        }
2854      { \__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.

```
2855      \bool_lazy_and:nnT
2856        { \bool_if_p:N \l__enumext_check_ans_bool }
2857        { \bool_not_p:n { \g__enumext_starred_bool } }
2858        {
2859          \bool_gset_true:N \g__enumext_check_ans_show_bool
2860          \tl_gset:NV \g__enumext_store_name_tl \l__enumext_store_name_tl
2861        }
```

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.

```
2862      \__enumext_after_stop_list:
2863      \__enumext_vspace_below:
2864      \bool_set_false:N \l__enumext_standar_bool
2865      \__enumext_resume_save_counter:
2866    }
```

(*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.

```
2867 \__enumext_after_env:nn {enumext}
2868   {
2869     \int_compare:nNnT { \l__enumext_level_int } = { 0 }
2870       {
2871         \bool_if:NT \g__enumext_check_ans_show_bool
2872           {
2873             \__enumext_check_ans_show:
2874           }
2875         \bool_gset_false:N \g__enumext_standar_bool
2876         \bool_gset_false:N \g__enumext_check_ans_show_bool
2877         \tl_gclear:N \g__enumext_store_name_tl
2878       }
2879   }
```

## 10.33   The environment keyans

The environment `keyans` also based on lists. The main differences with the `enumext` environment are the *nesting* and the way the *answers* (choice) will be stored and checked, this environment is intended exclusively for *"multiple choice questions"*.

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

```
2880 \NewDocumentEnvironment{keyans}{ O{} }
2881   {
2882     \__enumext_keyans_safe_exec:
2883     \__enumext_keyans_parse_keys:n {#1}
2884     \__enumext_before_list_v:
2885     \__enumext_start_list:nn
2886       { \tl_use:N \l__enumext_label_v_tl }
2887       {
2888         \__enumext_list_arg_two_v:
2889         \__enumext_before_keys_exec_v:
2890       }
2891     \__enumext_after_args_exec_v:
2892   }
2893   {
2894     \__enumext_check_starred_cmd:nn { item }{ keyans }
2895     \__enumext_stop_list:
2896     \__enumext_after_list_v:
2897   }
```

(*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.

```
2898  \cs_new_protected:Nn \__enumext_keyans_safe_exec:
2899    {
2900      \bool_if:NF \l__enumext_store_active_bool
2901        {
2902          \msg_error:nnnn { enumext } { wrong-place }{ keyans }{ save-ans }
2903        }
2904      \int_incr:N \l__enumext_keyans_level_int
2905      \bool_set_true:N \l__enumext_keyans_env_bool
2906      % Set false for interfering with enumext nested in keyans (yes, its possible and crayze)
2907      \bool_set_false:N \l__enumext_store_active_bool
2908      \int_compare:nNnT { \l__enumext_keyans_level_int } > { 1 }
2909        {
2910          \msg_error:nn { enumext } { keyans-nested }
2911        }
2912      \int_compare:nNnT { \l__enumext_level_int } > { 1 }
2913        {
2914          \msg_error:nn { enumext } { keyans-wrong-level }
2915        }
2916    }
```

(*End of definition for* \__enumext_keyans_safe_exec:.)

\__enumext_keyans_parse_keys:n

Parse [⟨*key = val*⟩] for `keyans` environment.

```
2917  \cs_new_protected:Npn \__enumext_keyans_parse_keys:n #1
2918    {
2919      \keys_set:nn { enumext / keyans } {#1}
2920    }
```

(*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.

```
2921  \cs_new_protected:Nn \__enumext_before_list_v:
2922    {
2923      \__enumext_vspace_above_v:
2924      \__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.

```
2925      \dim_compare:nNnT { \l__enumext_minipage_right_v_dim } > { \c_zero_dim }
2926        {
2927          \dim_set:Nn \l__enumext_minipage_left_v_dim
2928            {
2929              \linewidth - \l__enumext_minipage_right_v_dim - \l__enumext_minipage_hsep_v_dim
2930            }
```

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*.

```
2931          \bool_set_true:N \l__enumext_minipage_active_v_bool
2932          \int_gincr:N \g__enumext_minipage_stat_int
2933          \__enumext_keyans_mini_addvspace:
2934          \nointerlineskip\noindent
2935          \begin{__enumext_mini_env*}{ \l__enumext_minipage_left_v_dim }
2936        }
```

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

```
2937      \__enumext_keyans_multicols_start:
2938    }
```

(*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.

```
2939 \cs_new_protected:Nn \__enumext_keyans_multicols_start:
2940   {
2941     \int_compare:nNnT { \l__enumext_columns_v_int } > { 1 }
2942       {
```

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

```
2943         \dim_compare:nNnT { \l__enumext_columns_sep_v_dim } = { \c_zero_dim }
2944           {
2945             \dim_set:Nn \l__enumext_columns_sep_v_dim
2946               {
2947                 (
2948                   \l__enumext_labelwidth_v_dim + \l__enumext_labelsep_v_dim
2949                 ) / \l__enumext_columns_v_int
2950                 - \l__enumext_listoffset_v_dim
2951               }
2952           }
2953         \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).

```
2954         \skip_zero:N \multicolsep
2955         \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.

```
2956         \bool_if:NF \l__enumext_minipage_active_v_bool
2957           {
2958             \__enumext_keyans_multi_addvspace:
2959           }
2960         \raggedcolumns
2961         \begin{multicols}{ \l__enumext_columns_v_int }
2962       }
2963   }
```

(*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.

```
2964 \cs_new_protected:Nn \__enumext_keyans_multicols_stop:
2965   {
2966     \int_compare:nNnT { \l__enumext_columns_v_int } > { 1 }
2967       {
2968         \end{multicols}
2969         \bool_if:NF \l__enumext_minipage_active_v_bool
2970           {
2971             \par\addvspace{ \l__enumext_multicols_below_v_skip }
2972           }
2973       }
2974   }
```

(*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.

```
2975 \cs_new_protected:Nn \__enumext_after_list_v:
2976   {
2977     \bool_if:NTF \l__enumext_minipage_active_v_bool
2978       {
2979         \int_compare:nNnT { \g__enumext_minipage_stat_int } = { 1 }
2980           {
2981             \msg_warning:nn { enumext } { missing-miniright }
2982             \miniright
2983           }
2984         \int_gzero:N \g__enumext_minipage_stat_int
```

```
2985        \end{__enumext_mini_env*}
2986        \par\addvspace{ \l__enumext_minipage_after_skip }
2987      }
2988    { \__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.

```
2989      \bool_set_false:N \l__enumext_keyans_env_bool
2990      \__enumext_after_stop_list_v:
2991      \__enumext_vspace_below_v:
2992    }
```

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

### 10.34 The environment keyanspic and \anspic

The `keyanspic` environment is a list-based environment that uses the same configuration for *"spacing"* and ⟨*label*⟩ as the `keyans` environment, but it does not use `\item`.
The contents are passed to the environment by means of the `\anspic` command and are placed inside `minipage` environments, with the ⟨*label*⟩ underneath, adjusting widths according to the options passed to the environment.
Again it is necessary to "adjust" the spacing, both vertical and horizontal, to obtain an output like the one shown in the figure 12.
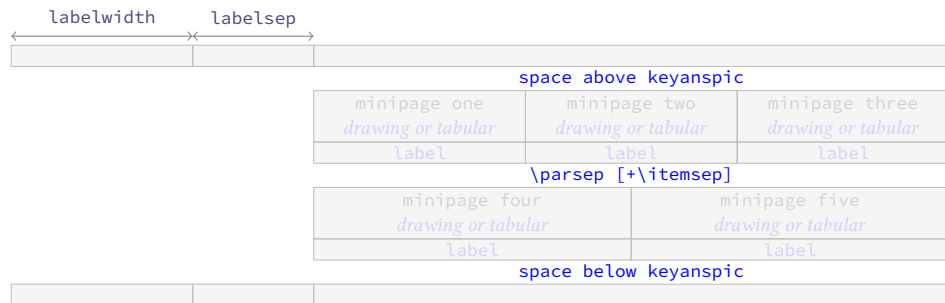


Figure 12: Representation of the `keyanspic` spacing in enumext.

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

#### 10.34.1 The command \anspic

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

```
2993    \NewDocumentCommand \anspic { s o +m }
2994      {
```

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

```
2995        \bool_if:NF \l__enumext_store_active_bool
2996          {
2997            \msg_error:nnnn { enumext } { wrong-place }{ keyanspic }{ save-ans }
2998          }
2999        \int_compare:nNnT { \l__enumext_level_int } > { 1 }
3000          {
3001            \msg_error:nn { enumext } { keyanspic-wrong-level }
3002          }
3003        \int_compare:nNnT { \l__enumext_keyans_level_int } = { 1 }
3004          {
3005            \msg_error:nnnn { enumext } { command-wrong-place }{ anspic }{ keyans }
3006          }
```

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.

```
3007        \seq_put_right:Nn \l__enumext_keyans_pic_body_seq
3008          {
3009            \__enumext_keyans_anspic_code:nnn { #1 } { #2 } { #3 }
3010          }
3011      }
```

(*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.

```
3012  \cs_new_protected:Nn \__enumext_keyans_anspic_code:nnn
3013    {
3014      \stepcounter { enumXvi }
3015      #3 \\
3016      \bool_if:nT { #1 }
3017        {
3018          \__enumext_keyans_addto_prop:n { #2 }
3019          \__enumext_keyans_store_ref:
3020          \__enumext_keyans_addto_seq:n { #2 }
3021          \int_incr:N \l__enumext_check_starred_cmd_int
3022          \bool_lazy_or:nnT
3023            { \bool_if_p:N \l__enumext_show_answer_bool }
3024            { \bool_if_p:N \l__enumext_show_position_bool }
3025            {
3026              \tl_set_eq:NN \l__enumext_label_v_tl \l__enumext_label_vi_tl
3027              \__enumext_keyans_show_left:n { #2 }
3028              \tl_set_eq:NN \l__enumext_label_vi_tl \l__enumext_label_v_tl
3029            }
3030        }
3031      \tl_use:N \l__enumext_label_font_style_v_tl
3032      \__enumext_wrapper_label_v:n { \l__enumext_label_vi_tl } \__enumext_keyans_show_item_opt:
3033    }
```

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

### 10.34.2 The environment keyanspic

keyanspic

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

```
3034  \NewDocumentEnvironment{keyanspic}{ o }
3035    {
3036      \__enumext_keyans_pic_safe_exec:
3037      \__enumext_start_list:nn
3038        { }
3039        {
3040          \__enumext_keyans_pic_arg_two:
3041        }
```

We apply the "adjusted" vertical spacing above the environment

```
3042      \vspace { \l__enumext_keyans_pic_above_skip }
3043    }
```

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.

```
3044    {
3045      \tl_if_novalue:nTF { #1 }
3046        {
3047          \__enumext_keyans_pic_do:e { \seq_count:N \l__enumext_keyans_pic_body_seq }
3048        }
3049        { \__enumext_keyans_pic_do:n { #1 } }
3050      \__enumext_stop_list:
3051      \__enumext_check_starred_cmd:nn { anspic } { keyanspic }
3052      \setcounter { enumXvi } { 0 }
3053      \vspace { \l__enumext_topsep_v_skip }
3054      %\bool_set_false:N \l__enumext_store_active_bool
3055    }
```

(*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.

```
3056  \cs_new_protected:Nn \__enumext_keyans_pic_safe_exec:
3057    {
3058      \int_incr:N \l__enumext_keyans_pic_level_int
3059      \int_compare:nNnT { \l__enumext_keyans_pic_level_int } > { 1 }
3060        {
3061          \msg_error:nn { enumext } { keyanspic-nested }
```

```
3062          }
3063     }
```

(*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.

```
3064 \cs_new_protected:Npn \__enumext_keyans_pic_skip_abs:N #1
3065   {
3066     \dim_compare:nNnT { #1 } < { 0pt }
3067       { \skip_set:Nn #1 { -#1 } }
3068   }
```

(*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.

```
3069 \cs_new_protected:Nn \__enumext_keyans_pic_arg_two:
3070   {
```

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.

```
3071     \bool_set_false:N \l__enumext_leftmargin_tmp_v_bool
3072     \__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.

```
3073     \skip_add:Nn \parsep { \itemsep }
3074     \dim_add:Nn  \leftmargin { -\labelwidth - \labelsep }
3075     \dim_zero:N  \labelwidth
3076     \dim_zero:N  \listparindent
3077     \dim_zero:N  \labelsep
3078     \skip_zero:N \partopsep
3079     \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.

```
3080     \__enumext_keyans_pic_skip_abs:N \parsep
3081     \skip_set:Nn \l__enumext_keyans_pic_above_skip
3082       {
3083         \box_dp:N \strutbox
3084         + \l__enumext_topsep_v_skip
3085         - \parsep
3086       }
3087     \__enumext_item_std:w \scan_stop:
3088   }
```

(*End of definition for \__enumext_keyans_pic_arg_two:.*)

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

```
3089 \cs_new_protected:Nn \__enumext_keyans_pic_do:n
3090   {
3091     \clist_map_function:nN { #1 } \__enumext_keyans_pic_row:n
3092   }
3093 \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.

```
3094 \cs_new_protected:Nn \__enumext_keyans_pic_row:n
3095   {
3096     \dim_set:Nn \l__enumext_keyans_pic_width_dim { \linewidth / #1 }
3097     \int_set:Nn \l__enumext_keyans_pic_above_int { \l__enumext_keyans_pic_below_int }
3098     \int_set:Nn \l__enumext_keyans_pic_below_int { \l__enumext_keyans_pic_above_int + #1 }
3099     \int_step_inline:nnn
```

```
3100          { \l__enumext_keyans_pic_above_int + 1 }
3101          { \l__enumext_keyans_pic_below_int }
3102          {
3103            \__enumext_minipage:w [ b ]{ \l__enumext_keyans_pic_width_dim }
3104              \centering
3105              \seq_item:Nn \l__enumext_keyans_pic_body_seq { ##1 }
3106            \__enumext_endminipage:
3107          }
3108        \par
3109      }
```

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

## 10.35  The environment enumext*

Generating horizontal list environments is NOT as simple as standard LATEX list environments. The fundamental part of the code is adapted from the shortlst package to a more modern version using expl3. It is not possible to redefine \item and \makelabel as in the non starred versions (at least I have not achieved it) and as we will make it behave differently, we have no other option than to define a cascade of functions.

To achieve the horizontal list environment we will capture the \item command and the content of this in an plain lrbox box using \makebox for the label and a minipage environment for the content passed to \item, we will also add the optional argument (⟨*number*⟩) to \item to be able to *join columns* horizontally, in simple terms, we want \item to behave in the same way as in the enumext environment but adding an optional first argument (⟨*number*⟩).

### 10.35.1  Functions for item box width

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

```
3110  \cs_new_protected:Nn \__enumext_starred_columns_set_vii:
3111    {
3112      \dim_compare:nNnT { \l__enumext_columns_sep_vii_dim } = { \c_zero_dim }
3113        {
3114          \dim_set:Nn \l__enumext_columns_sep_vii_dim
3115            {
3116              ( \l__enumext_labelwidth_vii_dim + \l__enumext_labelsep_vii_dim )
3117              / \l__enumext_columns_vii_int
3118            }
3119        }
3120      \int_set:Nn \l__enumext_tmpa_vii_int { \l__enumext_columns_vii_int - \c_one_int }
3121      \dim_set:Nn \l__enumext_item_width_vii_dim
3122        {
3123          ( \linewidth - \l__enumext_columns_sep_vii_dim * \l__enumext_tmpa_vii_int )
3124          / \l__enumext_columns_vii_int - \l__enumext_labelwidth_vii_dim
3125          - \l__enumext_labelsep_vii_dim
3126        }
3127      \dim_zero_new:N \itemwidth
3128    }
```

(*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.

```
3129  \cs_new_protected:Npn \__enumext_starred_joined_item_vii:n #1
3130    {
3131      \int_set:Nn \l__enumext_joined_item_vii_int {#1}
3132      \int_compare:nNnT { \l__enumext_joined_item_vii_int } > { \l__enumext_columns_vii_int }
3133        {
3134          \msg_warning:nnee { enumext } { item-joined }
3135            { \int_use:N \l__enumext_joined_item_vii_int }
3136            { \int_use:N \l__enumext_columns_vii_int }
3137          \int_set:Nn \l__enumext_joined_item_vii_int
3138            {
3139              \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + \c_one_int
3140            }
3141        }
3142      \int_compare:nNnT
3143        { \l__enumext_joined_item_vii_int }
3144          >
3145        { \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + \c_one_int }
3146        {
```

```
3147        \msg_warning:nnee { enumext } { item-joined-columns }
3148          { \int_use:N \l__enumext_joined_item_vii_int }
3149          {
3150            \int_eval:n
3151              { \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + \c_one_int }
3152          }
3153        \int_set:Nn \l__enumext_joined_item_vii_int
3154          {
3155            \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + \c_one_int
3156          }
3157      }
```

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

```
3158      \int_compare:nNnTF { \l__enumext_joined_item_vii_int } > { \c_one_int }
3159        {
3160          \int_set_eq:NN \l__enumext_joined_item_aux_vii_int \l__enumext_joined_item_vii_int
3161          \int_decr:N \l__enumext_joined_item_aux_vii_int
3162          \int_add:Nn \l__enumext_item_column_pos_vii_int { \l__enumext_joined_item_aux_vii_int }
3163          \int_gadd:Nn \g__enumext_item_count_all_vii_int { \l__enumext_joined_item_aux_vii_int }
3164          \dim_set:Nn \l__enumext_joined_width_vii_dim
3165            {
3166              \l__enumext_item_width_vii_dim * \l__enumext_joined_item_vii_int
3167              + (  \l__enumext_labelwidth_vii_dim + \l__enumext_labelsep_vii_dim
3168                + \l__enumext_columns_sep_vii_dim
3169              )*\l__enumext_joined_item_aux_vii_int
3170            }
3171          \dim_set_eq:NN \itemwidth \l__enumext_joined_width_vii_dim
3172        }
3173        {
3174          \dim_set_eq:NN \l__enumext_joined_width_vii_dim \l__enumext_item_width_vii_dim
3175          \dim_set_eq:NN \itemwidth \l__enumext_item_width_vii_dim
3176        }
3177    }
```

(*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.

```
3178  \cs_new_protected:Nn \__enumext_start_mini_vii:
3179    {
3180      \dim_compare:nNnT { \l__enumext_minipage_right_vii_dim } > { \c_zero_dim }
3181        {
3182          \dim_set:Nn \l__enumext_minipage_left_vii_dim
3183            {
3184              \linewidth
3185              - \l__enumext_minipage_right_vii_dim
3186              - \l__enumext_minipage_hsep_vii_dim
3187            }
3188          \bool_set_true:N \l__enumext_minipage_active_vii_bool
3189          \dim_gset_eq:NN
3190            \g__enumext_minipage_right_vii_dim
3191            \l__enumext_minipage_right_vii_dim
3192          \__enumext_mini_addvspace_vii:
3193          \nointerlineskip\noindent
3194          \begin{__enumext_mini_env*}{ \l__enumext_minipage_left_vii_dim }
3195        }
3196    }
```

(*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"*.

```
3197  \cs_new_protected:Nn \__enumext_stop_mini_vii:
3198    {
3199      \bool_if:NT \l__enumext_minipage_active_vii_bool
3200        {
```

```
3201          \end{__enumext_mini_env*}
3202          \hfill
3203          \bool_gset_true:N \g__enumext_minipage_active_vii_bool
3204        }
3205      }
```

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"*.

```
3206  \__enumext_after_env:nn {enumext*}
3207    {
3208      \bool_if:NT \g__enumext_minipage_active_vii_bool
3209        {
3210          \begin{__enumext_mini_env*}{ \g__enumext_minipage_right_vii_dim }
3211            \par\addvspace { \g__enumext_minipage_right_skip }
3212            \bool_if:NF \g__enumext_minipage_center_vii_bool
3213              {
3214                \centering
3215              }
3216            \tl_use:N \g__enumext_miniright_code_vii_tl % the code
3217          \end{__enumext_mini_env*}
3218          \par\addvspace{ \g__enumext_minipage_after_skip }
3219        }
3220      \bool_gset_false:N \g__enumext_minipage_active_vii_bool
3221      \bool_gset_true:N \g__enumext_minipage_center_vii_bool
3222      \tl_gclear:N \g__enumext_miniright_code_vii_tl
3223      \dim_gzero:N \g__enumext_minipage_right_vii_dim
3224      \bool_gset_false:N \g__enumext_starred_bool
3225    }
```

(*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.

```
3226  \NewDocumentEnvironment{enumext*}{ o }
3227    {
3228      \__enumext_safe_exec_vii:
3229      \__enumext_parse_keys_vii:n {#1}
3230      \__enumext_before_list_vii:
3231      \__enumext_start_store_level_vii:
3232      \__enumext_start_list:nn { }
3233        {
3234          \__enumext_list_arg_two_vii:
3235          \__enumext_before_keys_exec_vii:
3236        }
3237      \__enumext_starred_columns_set_vii:
3238      \item[] \scan_stop:
3239      \cs_set_eq:NN \__enumext_stop_item_tmp_vii: \noindent
3240      \cs_set_eq:NN \item \__enumext_start_item_tmp_vii:
3241    }
3242    {
3243      \__enumext_stop_item_tmp_vii:
3244      \__enumext_remove_extra_parsep_vii:
3245      \__enumext_stop_list:
3246      \__enumext_stop_store_level_vii:
3247      \__enumext_after_list_vii:
3248    }
```

(*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`.

```
3249  \cs_new_protected:Nn \__enumext_safe_exec_vii:
3250    {
3251      \__enumext_current_env_set_bool:
3252      \int_incr:N \l__enumext_level_h_int
3253      \int_compare:nNnT { \l__enumext_level_h_int } > { 1 }
3254        {
3255          \msg_error:nn { enumext } { nested }
3256        }
3257      \bool_set_true:N \l__enumext_starred_bool
```

```
3258        \__enumext_check_first_level:
3259      }
```

(*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.

```
3260  \cs_new_protected:Npn \__enumext_parse_keys_vii:n #1
3261    {
3262      \tl_if_novalue:nF {#1}
3263        {
3264          \str_clear:N \l__enumext_series_str
3265          \keys_set:nn { enumext / enumext* } {#1}
3266          \__enumext_parse_series:n {#1}
3267          \bool_if:NT \l__enumext_store_active_bool
3268            {
3269              \__enumext_parse_store_keys_vii:n {#1}
3270            }
3271        }
3272    }
```

(*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:.

```
3273  \cs_new_protected:Npn \__enumext_parse_store_keys_vii:n #1
3274    {
3275      \bool_if:NF \l__enumext_store_columns_vii_bool
3276        {
3277          \regex_match:nnT { \b columns\b } {#1}
3278            {
3279              \int_set_eq:NN
3280                \l__enumext_store_columns_vii_int
3281                \l__enumext_columns_vii_int
3282              \tl_put_right:Ne \l__enumext_store_opt_vii_tl
3283                {
3284                  columns = \exp_not:V \l__enumext_store_columns_vii_int ,
3285                }
3286            }
3287        }
3288      \bool_if:NF \l__enumext_store_columns_sep_vii_bool
3289        {
3290          \regex_match:nnT { \b columns-sep \b} {#1}
3291            {
3292              \dim_set_eq:NN
3293                \l__enumext_store_columns_sep_vii_dim
3294                \l__enumext_columns_sep_vii_dim
3295              \tl_put_right:Ne \l__enumext_store_opt_vii_tl
3296                {
3297                  columns-sep = \exp_not:V \l__enumext_store_columns_sep_vii_dim,
3298                }
3299            }
3300        }
3301    }
```

(*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.

```
3302  \cs_new_protected:Nn \__enumext_before_list_vii:
3303    {
3304      \__enumext_vspace_above_vii:
3305      \__enumext_check_ans_exec: % need by chek-ans
3306      \__enumext_before_args_exec_vii:
3307      \__enumext_start_mini_vii:
3308    }
```

(*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.

```
3309  \cs_new_protected:Nn \__enumext_after_list_vii:
3310    {
3311      \__enumext_stop_mini_vii:
3312      \__enumext_after_stop_list_vii:
3313      \__enumext_vspace_below_vii:
3314      \bool_set_false:N \l__enumext_starred_bool
3315      \__enumext_resume_save_counter:
3316    }
```

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

`\__enumext_start_store_level_vii:`
`\__enumext_stop_store_level_vii:`

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

```
3317  \cs_new_protected:Nn \__enumext_start_store_level_vii:
3318    {
3319      \bool_if:NT \l__enumext_store_active_bool
3320        {
3321          \int_compare:nNnT { \l__enumext_level_int } > { \c_zero_int }
3322            {
3323              \__enumext_store_level_open_vii:
3324            }
3325        }
3326    }
3327  \cs_new_protected:Nn \__enumext_stop_store_level_vii:
3328    {
3329      \bool_if:NT \l__enumext_store_active_bool
3330        {
3331          \int_compare:nNnT { \l__enumext_level_int } > { \c_zero_int }
3332            {
3333              \__enumext_store_level_close_vii:
3334            }
3335        }
3336    }
```

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

### 10.35.2 The command \item in enumext*

`\__enumext_start_item_tmp_vii:`

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

```
3337  \cs_new_protected_nopar:Nn \__enumext_start_item_tmp_vii:
3338    {
3339      \__enumext_stop_item_tmp_vii:
3340      \int_incr:N \l__enumext_item_column_pos_vii_int
3341      \int_gincr:N \g__enumext_item_count_all_vii_int
3342      \__enumext_item_peek_args_vii:
3343    }
```

(*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).

```
3344  \cs_new_protected:Nn \__enumext_item_peek_args_vii:
3345    {
3346      \peek_meaning:NTF (
3347        { \__enumext_joined_item_vii:w }
3348        { \__enumext_joined_item_vii:w (1) }
3349    }
```

*(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.

```
3350 \cs_new_protected:Npn \__enumext_joined_item_vii:w (#1)
3351   {
3352     \__enumext_starred_joined_item_vii:n {#1}
3353     \peek_meaning_remove:NTF *
3354       { \__enumext_starred_item_vii:w  }
3355       { \__enumext_standard_item_vii:w }
3356   }
```

*(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 ].

```
3357 \cs_new_protected:Npn \__enumext_standard_item_vii:w
3358   {
3359     \bool_set_false:N \l__enumext_item_starred_vii_bool
3360     \peek_meaning:NTF [
3361       {
3362         \bool_set_eq:NN
3363           \l__enumext_wrap_label_vii_bool
3364           \l__enumext_wrap_label_opt_vii_bool
3365         \__enumext_start_item_vii:w
3366       }
3367       {
3368         \bool_set_true:N \l__enumext_wrap_label_vii_bool
3369         \legacy_if_set_true:n { @noitemarg }
3370         \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
3371       }
3372   }
```

*(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*⟩].

```
3373 \cs_new_protected:Npn \__enumext_starred_item_vii:w
3374   {
3375     \bool_set_true:N \l__enumext_item_starred_vii_bool
3376     \bool_set_true:N \l__enumext_wrap_label_vii_bool
3377     \peek_meaning:NTF [
3378       { \__enumext_starred_item_vii_aux_i:w }
3379       { \__enumext_starred_item_vii_aux_ii:w }
3380   }
3381 \cs_new_protected:Npn \__enumext_starred_item_vii_aux_i:w [#1]
3382   {
3383     \tl_gset:Nn \g__enumext_item_symbol_aux_vii_tl {#1}
3384     \__enumext_starred_item_vii_aux_ii:w
3385   }
3386 \cs_new_protected:Npn \__enumext_starred_item_vii_aux_ii:w
3387   {
3388     \peek_meaning:NTF [
3389       { \__enumext_starred_item_vii_aux_iii:w }
3390       {
3391         \dim_set_eq:NN
3392           \l__enumext_item_symbol_sep_vii_dim
3393           \l__enumext_labelsep_vii_dim
3394         \legacy_if_set_true:n { @noitemarg }
3395         \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
3396       }
3397   }
3398 \cs_new_protected:Npn \__enumext_starred_item_vii_aux_iii:w [#1]
```

```
3399        {
3400          \dim_set:Nn \l__enumext_item_symbol_sep_vii_dim {#1}
3401          \legacy_if_set_true:n { @noitemarg }
3402          \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
3403        }
```

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

### 10.35.3 Real definition of \item in enumext*

\__enumext_start_item_vii:w

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

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

```
3404  \cs_new_protected_nopar:Npn \__enumext_start_item_vii:w [#1]
3405    {
3406      \cs_set_eq:NN \__enumext_stop_item_tmp_vii: \__enumext_stop_item_vii:
3407      \legacy_if:nT { @noitemarg }
3408        {
3409          \legacy_if_set_false:n { @noitemarg }
3410          \legacy_if:nT { @nmbrlist }
3411            {
3412              \bool_if:NT \l__enumext_hyperref_bool
3413                {
3414                  \legacy_if_set_true:n { @hyper@item }
3415                }
3416              \refstepcounter{enumXvii}
3417              \bool_if:NT \l__enumext_check_ans_bool
3418                {
3419                  \int_gincr:N \g__enumext_count_item_number_int
3420                }
3421            }
3422        }
```

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.

```
3423        \group_begin:
3424          \lrbox{ \l__enumext_item_text_vii_box }
3425            \bool_if:NF \l__enumext_footnotes_key_bool
3426              {
3427                \__enumext_renew_footnote:
3428              }
3429            \bool_if:NT \l__enumext_item_starred_vii_bool
3430              {
3431                \tl_if_blank:VT \g__enumext_item_symbol_aux_vii_tl
3432                  {
3433                    \tl_gset_eq:NN
3434                      \g__enumext_item_symbol_aux_vii_tl \l__enumext_item_symbol_vii_tl
3435                  }
3436                \mode_leave_vertical:
3437                \skip_horizontal:n { -\l__enumext_item_symbol_sep_vii_dim }
3438                \makebox[ 0pt ][ r ]{ \g__enumext_item_symbol_aux_vii_tl }
3439                \skip_horizontal:N \l__enumext_item_symbol_sep_vii_dim
3440                \tl_gclear:N \g__enumext_item_symbol_aux_vii_tl
3441              }
3442            \group_begin:
3443              \tl_use:N \l__enumext_label_font_style_vii_tl
3444              \bool_if:NTF \l__enumext_wrap_label_vii_bool
3445                {
3446                  \makebox[ \l__enumext_labelwidth_vii_dim ][ \l__enumext_align_label_vii_str ]
3447                    { \__enumext_wrapper_label_vii:n {#1} }
3448                }
3449                {
```

```
3450              \makebox[ \l__enumext_labelwidth_vii_dim ][ \l__enumext_align_label_vii_str ]{ #1 }
3451            }
3452          \group_end:
3453          \skip_horizontal:N \l__enumext_labelsep_vii_dim
3454          \tl_use:N \l__enumext_after_list_args_vii_tl
3455          \__enumext_minipage:w [ t ]{ \l__enumext_joined_width_vii_dim  }
3456            \skip_set_eq:NN \parindent \l__enumext_listparindent_vii_dim
3457            \skip_set_eq:NN \parskip \l__enumext_parsep_vii_skip
3458            \tl_use:N \l__enumext_fake_item_indent_vii_tl
3459        }
```

(*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.

```
3460 \cs_new_protected_nopar:Nn \__enumext_stop_item_vii:
3461   {
3462          \__enumext_endminipage:
3463        \endlrbox
3464      \group_end:
3465      \box_set_wd:Nn \l__enumext_item_text_vii_box
3466        {
3467          \l__enumext_joined_width_vii_dim
3468          + \l__enumext_labelwidth_vii_dim
3469          + \l__enumext_labelsep_vii_dim
3470        }
3471      \int_set:Nn \hbadness { 10000 }
3472      \box_use:N \l__enumext_item_text_vii_box
3473      \bool_if:NF \l__enumext_footnotes_key_bool
3474        {
3475          \__enumext_print_footnote:
3476        }
3477      \int_compare:nNnTF { \l__enumext_item_column_pos_vii_int } = { \l__enumext_columns_vii_int }
3478        {
3479          \par\noindent
3480          \int_zero:N \l__enumext_item_column_pos_vii_int
3481        }
3482        { \hspace{ \l__enumext_columns_sep_vii_dim } }
3483    }
```

(*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.

```
3484 \cs_new_protected:Nn \__enumext_remove_extra_parsep_vii:
3485   {
3486      \int_compare:nNnT
3487        {
3488          \int_mod:nn {  \g__enumext_item_count_all_vii_int } { \l__enumext_columns_vii_int }
3489        }
3490        =
3491        { \c_zero_int }
3492        {
3493          \par
3494          \vspace{ -\l__enumext_itemsep_vii_skip }
3495          \int_gzero:N \g__enumext_item_count_all_vii_int
3496        }
3497    }
```

(*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.

```
3498 \__enumext_after_env:nn {enumext*}
3499   {
3500      \int_compare:nNnT { \l__enumext_level_int } = { 0 }
3501        {
3502          \bool_if:NT \g__enumext_check_ans_show_h_bool
3503            {
3504              \__enumext_check_ans_show:
```

```
3505          }
3506          \bool_gset_false:N \g__enumext_starred_bool
3507          \bool_gset_false:N \g__enumext_check_ans_show_h_bool
3508          \tl_gclear:N \g__enumext_store_name_tl
3509        }
3510     }
```

### 10.36 The environment keyans*

#### 10.36.1 Functions for item box width

\__enumext_starred_columns_set_viii:

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

```
3511   \cs_new_protected:Nn \__enumext_starred_columns_set_viii:
3512     {
3513       \dim_compare:nNnT { \l__enumext_columns_sep_viii_dim } = { \c_zero_dim }
3514         {
3515           \dim_set:Nn \l__enumext_columns_sep_viii_dim
3516             {
3517               ( \l__enumext_labelwidth_viii_dim + \l__enumext_labelsep_viii_dim )
3518               / \l__enumext_columns_viii_int
3519             }
3520         }
3521       \int_set:Nn \l__enumext_tmpa_viii_int { \l__enumext_columns_viii_int - \c_one_int }
3522       \dim_set:Nn \l__enumext_item_width_viii_dim
3523         {
3524           ( \linewidth - \l__enumext_columns_sep_viii_dim * \l__enumext_tmpa_viii_int )
3525           / \l__enumext_columns_viii_int - \l__enumext_labelwidth_viii_dim
3526           - \l__enumext_labelsep_viii_dim
3527         }
3528       \dim_zero_new:N \itemwidth
3529     }
```

(*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.

```
3530   \cs_new_protected:Npn \__enumext_starred_joined_item_viii:n #1
3531     {
3532       \int_set:Nn \l__enumext_joined_item_viii_int {#1}
3533       \int_compare:nNnT { \l__enumext_joined_item_viii_int } > { \l__enumext_columns_viii_int }
3534         {
3535           \msg_warning:nnee { enumext } { item-joined }
3536             { \int_use:N \l__enumext_joined_item_viii_int }
3537             { \int_use:N \l__enumext_columns_viii_int }
3538           \int_set:Nn \l__enumext_joined_item_viii_int
3539             {
3540               \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + \c_one_int
3541             }
3542         }
3543       \int_compare:nNnT
3544         { \l__enumext_joined_item_viii_int }
3545         >
3546         { \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + \c_one_int }
3547         {
3548           \msg_warning:nnee { enumext } { item-joined-columns }
3549             { \int_use:N \l__enumext_joined_item_viii_int }
3550             {
3551               \int_eval:n
3552                 { \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + \c_one_int }
3553             }
3554           \int_set:Nn \l__enumext_joined_item_viii_int
3555             {
3556               \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + \c_one_int
3557             }
3558         }
3559       \int_compare:nNnTF { \l__enumext_joined_item_viii_int } > { \c_one_int }
3560         {
3561           \int_set_eq:NN \l__enumext_joined_item_aux_viii_int \l__enumext_joined_item_viii_int
3562           \int_decr:N \l__enumext_joined_item_aux_viii_int
3563           \int_add:Nn \l__enumext_item_column_pos_viii_int { \l__enumext_joined_item_aux_viii_int }
3564           \int_gadd:Nn \g__enumext_item_count_all_viii_int { \l__enumext_joined_item_aux_viii_int }
```

```
3565        \dim_set:Nn \l__enumext_joined_width_viii_dim
3566          {
3567            \l__enumext_item_width_viii_dim * \l__enumext_joined_item_viii_int
3568            + (  \l__enumext_labelwidth_viii_dim + \l__enumext_labelsep_viii_dim
3569              + \l__enumext_columns_sep_viii_dim
3570            )*\l__enumext_joined_item_aux_viii_int
3571          }
3572        \dim_set_eq:NN \itemwidth \l__enumext_joined_width_viii_dim
3573      }
3574      {
3575        \dim_set_eq:NN \l__enumext_joined_width_viii_dim \l__enumext_item_width_viii_dim
3576        \dim_set_eq:NN \itemwidth \l__enumext_item_width_viii_dim
3577      }
3578    }
```

(*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.

```
3579 \cs_new_protected:Nn \__enumext_start_mini_viii:
3580    {
3581      \dim_compare:nNnT { \l__enumext_minipage_right_viii_dim } > { \c_zero_dim }
3582        {
3583          \dim_set:Nn \l__enumext_minipage_left_viii_dim
3584            {
3585              \linewidth
3586              - \l__enumext_minipage_right_viii_dim
3587              - \l__enumext_minipage_hsep_viii_dim
3588            }
3589          \bool_set_true:N \l__enumext_minipage_active_viii_bool
3590          \dim_gset_eq:NN
3591            \g__enumext_minipage_right_viii_dim
3592            \l__enumext_minipage_right_viii_dim
3593          \__enumext_mini_addvspace_viii:
3594          \nointerlineskip\noindent
3595          \begin{__enumext_mini_env*}{ \l__enumext_minipage_left_viii_dim }
3596        }
3597    }
3598 \cs_new_protected:Nn \__enumext_stop_mini_viii:
3599    {
3600      \bool_if:NT \l__enumext_minipage_active_viii_bool
3601        {
3602          \end{__enumext_mini_env*}
3603          \hfill
3604          \bool_gset_true:N \g__enumext_minipage_active_viii_bool
3605        }
3606    }
3607 \__enumext_after_env:nn {keyans*}
3608    {
3609      \bool_if:NT \g__enumext_minipage_active_viii_bool
3610        {
3611          \begin{__enumext_mini_env*}{ \g__enumext_minipage_right_viii_dim }
3612            \par\addvspace { \g__enumext_minipage_right_skip }
3613            \bool_if:NF \g__enumext_minipage_center_viii_bool
3614              {
3615                \centering
3616              }
3617            \tl_use:N \g__enumext_miniright_code_viii_tl % the code
3618          \end{__enumext_mini_env*}
3619          \par\addvspace{ \g__enumext_minipage_after_skip }
3620        }
3621      \bool_gset_false:N \g__enumext_minipage_active_viii_bool
3622      \bool_gset_true:N \g__enumext_minipage_center_viii_bool
3623      \tl_gclear:N \g__enumext_miniright_code_viii_tl
3624      \dim_gzero:N \g__enumext_minipage_right_viii_dim
3625    }
```

(*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.

```
3626  \NewDocumentEnvironment{keyans*}{ o }
3627    {
3628      \__enumext_safe_exec_viii:
3629      \__enumext_parse_keys_viii:n {#1}
3630      \__enumext_before_list_viii:
3631      \__enumext_start_list:nn { }
3632        {
3633          \__enumext_list_arg_two_viii:
3634          \__enumext_before_keys_exec_viii:
3635        }
3636      \__enumext_starred_columns_set_viii:
3637      \item[] \scan_stop:
3638      \cs_set_eq:NN \__enumext_stop_item_tmp_viii: \noindent
3639      \cs_set_eq:NN \item \__enumext_start_item_tmp_viii:
3640    }
3641    {
3642      \__enumext_stop_item_tmp_viii:
3643      \__enumext_remove_extra_parsep_viii:
3644      \__enumext_check_starred_cmd:nn { item }{ keyans* }
3645      \__enumext_stop_list:
3646      \__enumext_after_list_viii:
3647    }
```

(*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.

```
3648  \cs_new_protected:Nn \__enumext_safe_exec_viii:
3649    {
3650      \int_incr:N \l__enumext_keyans_level_h_int
3651      \int_compare:nNnT { \l__enumext_keyans_level_h_int } > { 1 }
3652        {
3653          \msg_error:nn { enumext } { nested }
3654        }
3655      % Set false for interfering with enumext nested in keyans* (yes, its possible and crayze)
3656      \bool_set_false:N \l__enumext_store_active_bool
3657      \int_compare:nNnT { \l__enumext_level_int } > { 1 }
3658        {
3659          \msg_error:nn { enumext } { keyans-wrong-level }
3660        }
3661    }
```

(*End of definition for* \_\_enumext_safe_exec_viii:.)

\_\_enumext_parse_keys_viii:n    Parse [⟨*key* = *val*⟩] for keyans*.

```
3662  \cs_new_protected:Npn \__enumext_parse_keys_viii:n #1
3663    {
3664      \tl_if_novalue:nF {#1}
3665        {
3666          \keys_set:nn { enumext / keyans* } {#1}
3667        }
3668    }
```

(*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.

```
3669  \cs_new_protected:Nn \__enumext_before_list_viii:
3670    {
3671      \__enumext_vspace_above_viii:
3672      \__enumext_before_args_exec_viii:
3673      \__enumext_start_mini_viii:
3674    }
```

(*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.

```
3675  \cs_new_protected:Nn \__enumext_after_list_viii:
3676    {
```

```
3677        \__enumext_stop_mini_viii:
3678        \__enumext_after_stop_list_viii:
3679        \__enumext_vspace_below_viii:
3680      }
```

(*End of definition for* \__enumext_after_list_viii:.)

### 10.36.2   The command \item in keyans*

The idea here is to make the \item command behave in the same way as in the keyans environment with the difference of the optional argument (⟨*number*⟩) which works in the same way as in the enumext* environment. In simple terms we want to store the ⟨*label*⟩ next to the [⟨*content*⟩] if it is present in the ⟨*sequence*⟩ and ⟨*prop list*⟩ defined by save-ans key for \item*, \item*[⟨*content*⟩], \item(⟨*number*⟩)* and \item(⟨*number*⟩)*[⟨*content*⟩] commands.

\__enumext_start_item_tmp_viii:  First we will call the function \__enumext_stop_item_tmp_viii: that we will redefine later, we will increment the value of \l__enumext_item_column_pos_viii_int that will count the item's by rows and the value of \g__enumext_item_count_all_viii_int that will count the total of item's in the environment. After that we will call the function \__enumext_item_peek_args_viii: that will handle the arguments passed to \item.

```
3681  \cs_new_protected_nopar:Nn \__enumext_start_item_tmp_viii:
3682    {
3683      \__enumext_stop_item_tmp_viii:
3684      \int_incr:N \l__enumext_item_column_pos_viii_int
3685      \int_gincr:N \g__enumext_item_count_all_viii_int
3686      \__enumext_item_peek_args_viii:
3687    }
```

(*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).

```
3688  \cs_new_protected:Nn \__enumext_item_peek_args_viii:
3689    {
3690      \peek_meaning:NTF (
3691        { \__enumext_joined_item_viii:w }
3692        { \__enumext_joined_item_viii:w (1) }
3693    }
```

(*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.

```
3694  \cs_new_protected:Npn \__enumext_joined_item_viii:w (#1)
3695    {
3696      \__enumext_starred_joined_item_viii:n {#1}
3697      \peek_meaning_remove:NTF *
3698        { \__enumext_starred_item_viii:w  }
3699        { \__enumext_standard_item_viii:w }
3700    }
```

(*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 ].

```
3701  \cs_new_protected:Npn \__enumext_standard_item_viii:w
3702    {
3703      \bool_set_false:N \l__enumext_item_starred_viii_bool
3704        \peek_meaning:NTF [
3705          {
```

```
3706          \bool_set_eq:NN
3707            \l__enumext_wrap_label_viii_bool
3708            \l__enumext_wrap_label_opt_viii_bool
3709          \__enumext_start_item_viii:w
3710        }
3711        {
3712          \bool_set_true:N \l__enumext_wrap_label_viii_bool
3713          \legacy_if_set_true:n { @noitemarg }
3714          \__enumext_start_item_viii:w [ \l__enumext_label_viii_tl ]
3715        }
3716    }
```

(*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*⟩].

```
3717 \cs_new_protected:Npn \__enumext_starred_item_viii:w
3718    {
3719      \bool_set_true:N \l__enumext_item_starred_viii_bool
3720      \bool_set_true:N \l__enumext_wrap_label_viii_bool
3721      \peek_meaning:NTF [
3722        { \__enumext_starred_item_viii_aux_i:w }
3723        { \__enumext_starred_item_viii_aux_ii:w }
3724    }
```

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*⟩.

```
3725 \cs_new_protected:Npn \__enumext_starred_item_viii_aux_i:w [#1]
3726    {
3727      \tl_clear:N \l__enumext_store_keyans_label_tl
3728      \tl_if_novalue:nF { #1 }
3729        {
3730          \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_tl
3731            {
3732              \tl_put_right:Ne \l__enumext_store_keyans_label_tl { \l__enumext_store_keyans_item_op
3733              \tl_put_right:Ne \l__enumext_store_keyans_label_tl { #1 }
3734            }
3735          \tl_set:Ne \l__enumext_keyans_item_opt_tl { #1 }
3736        }
3737      \__enumext_starred_item_viii_aux_ii:w
3738    }
3739 \cs_new_protected:Npn \__enumext_starred_item_viii_aux_ii:w
3740    {
3741      \legacy_if_set_true:n { @noitemarg }
3742      \__enumext_start_item_viii:w [ \l__enumext_label_viii_tl ]
3743    }
```

(*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.

```
3744 \cs_new_protected:Nn \__enumext_starred_item_exec:
3745    {
3746      \tl_put_left:Ne \l__enumext_store_keyans_label_tl { \l__enumext_label_viii_tl }
3747      \__enumext_store_addto_prop:V \l__enumext_store_keyans_label_tl
3748      \__enumext_keyans_store_ref:
3749      \tl_put_left:Ne \l__enumext_store_keyans_label_tl { \item }
3750      \__enumext_keyans_addto_seq_link:
3751      \int_incr:N \l__enumext_check_starred_cmd_int
3752      \bool_if:NT \l__enumext_show_answer_bool
3753        {
3754          \__enumext_print_keyans_box:NN \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
3755        }
3756      \bool_if:NT \l__enumext_show_position_bool
3757        {
3758          \tl_set:Ne \l__enumext_mark_answer_sym_tl
3759            {
```

```
3760                \group_begin:
3761                  \exp_not:N \normalfont
3762                  \exp_not:N \footnotesize [ \int_eval:n
3763                    {
3764                        \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
3765                    }
3766                    ]
3767                \group_end:
3768              }
3769            \__enumext_print_keyans_box:NN \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
3770          }
3771      }
```

(*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.

```
3772 \cs_new_protected_nopar:Npn \__enumext_start_item_viii:w [#1]
3773    {
3774      \cs_set_eq:NN \__enumext_stop_item_tmp_viii: \__enumext_stop_item_viii:
3775      \legacy_if:nT { @noitemarg }
3776        {
3777            \legacy_if_set_false:n { @noitemarg }
3778            \legacy_if:nT { @nmbrlist }
3779              {
3780                \bool_if:NT \l__enumext_hyperref_bool
3781                  {
3782                      \legacy_if_set_true:n { @hyper@item }
3783                  }
3784                \refstepcounter{enumXviii}
3785              }
3786        }
```

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

```
3787        \group_begin:
3788          \lrbox{ \l__enumext_item_text_viii_box }
3789            \bool_if:NF \l__enumext_footnotes_key_bool
3790              {
3791                  \__enumext_renew_footnote:
3792              }
3793            \bool_if:NT \l__enumext_item_starred_viii_bool
3794              {
3795                  \__enumext_starred_item_exec:
3796              }
3797            \group_begin:
3798              \tl_use:N \l__enumext_label_font_style_viii_tl
3799              \bool_if:NTF \l__enumext_wrap_label_viii_bool
3800                {
3801                  \makebox[ \l__enumext_labelwidth_viii_dim ][ \l__enumext_align_label_viii_str ]
3802                    { \__enumext_wrapper_label_viii:n {#1} }
3803                }
3804                {
3805                  \makebox[ \l__enumext_labelwidth_viii_dim ][ \l__enumext_align_label_viii_str ]{ #1
3806                }
3807            \group_end:
3808            \skip_horizontal:N \l__enumext_labelsep_viii_dim
3809            \tl_use:N \l__enumext_after_list_args_viii_tl
3810            \__enumext_minipage:w [ t ]{ \l__enumext_joined_width_viii_dim  }
3811              \skip_set_eq:NN \parindent \l__enumext_listparindent_viii_dim
3812              \skip_set_eq:NN \parskip \l__enumext_parsep_viii_skip
3813              \bool_if:NT \l__enumext_item_starred_viii_bool
3814                {
3815                  \tl_use:N \l__enumext_fake_item_indent_viii_tl
3816                  \__enumext_keyans_show_item_opt: \skip_horizontal:n { -\l__enumext_fake_item_indent_
3817                }
3818                {
3819                  \tl_use:N \l__enumext_fake_item_indent_viii_tl
3820                }
3821      }
```

(*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.

```
3822  \cs_new_protected_nopar:Nn \__enumext_stop_item_viii:
3823    {
3824        \__enumext_endminipage:
3825      \endlrbox
3826    \group_end:
3827    \box_set_wd:Nn \l__enumext_item_text_viii_box
3828      {
3829        \l__enumext_joined_width_viii_dim
3830        + \l__enumext_labelwidth_viii_dim
3831        + \l__enumext_labelsep_viii_dim
3832      }
3833    \int_set:Nn \hbadness { 10000 }
3834    \box_use:N \l__enumext_item_text_viii_box
3835    \bool_if:NF \l__enumext_footnotes_key_bool
3836      {
3837        \__enumext_print_footnote:
3838      }
3839    \int_compare:nNnTF { \l__enumext_item_column_pos_viii_int } = { \l__enumext_columns_viii_int }
3840      {
3841        \par\noindent
3842        \int_zero:N \l__enumext_item_column_pos_viii_int
3843      }
3844      { \hspace{ \l__enumext_columns_sep_viii_dim } }
3845    }
```

(*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.

```
3846  \cs_new_protected:Nn \__enumext_remove_extra_parsep_viii:
3847    {
3848    \int_compare:nNnT
3849      {
3850        \int_mod:nn {  \g__enumext_item_count_all_viii_int } { \l__enumext_columns_viii_int }
3851      }
3852      =
3853      { \c_zero_int }
3854      {
3855        \par
3856        \vspace{ -\l__enumext_itemsep_viii_skip }
3857        \int_gzero:N \g__enumext_item_count_all_viii_int
3858      }
3859    }
```

(*End of definition for* \__enumext_remove_extra_parsep_viii:.)

### 10.37   The command \getkeyans

\getkeyans

The \getkeyans command takes a mandatory argument of the form {⟨*store name : position*⟩}. Retrieve a "*single*" content stored by \anskey, \anspic* and \item* from ⟨*prop list*⟩ defined by save-ans key.

```
3860  \NewDocumentCommand \getkeyans { m }
3861    {
3862      \exp_args:Ne \__enumext_getkeyans_aux:n
3863        { \tl_to_str:e { \text_expand:n {#1} } }
3864    }
```

(*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.

```
3865  \cs_new_protected:Npn \__enumext_getkeyans_aux:n #1
3866    {
3867    \str_if_in:nnTF {#1} { : }
3868      {
3869        \use:e
3870          {
3871            \cs_set:Npn \exp_not:N \__enumext_tmp:w ##1 \c_colon_str ##2 \scan_stop:
3872              { {##1} {##2} }
```

```
3873            }
3874          \exp_after:wN \__enumext_getkeyans:nn \__enumext_tmp:w #1 \scan_stop:
3875        }
3876      { \msg_error:nnn { enumext } { missing-colon } {#1} }
3877    }
```

(*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*⟩.

```
3878  \cs_new_protected:Npn \__enumext_getkeyans:nn #1 #2
3879    {
3880      \prop_if_exist:cF { g__enumext_#1_prop }
3881        { \msg_error:nnn { enumext } { undefined-storage-anskey } {#1} }
3882      \group_begin:
3883        \prop_item:cn { g__enumext_#1_prop }{#2}
3884      \group_end:
3885    }
```

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

### 10.38   The command \printkeyans

The \printkeyans command prints *"all stored content"* in the ⟨*sequence*⟩ defined by the save-ans key. The first thing we will do is to define a set of ⟨*keys*⟩ with which we will control the options of the different nesting levels for the enumext and enumext* environment by storing the values of these in the token list variables \l__enumext_print_keyans_X_tl.

```
3886  \keys_define:nn  { keyanskey / print }
3887    {
3888      level-1 .code:n      = \tl_put_right:Nn \l__enumext_print_keyans_i_tl
3889                              {
3890                                \setenumext[level,1] {#1} \setenumext[print,1] {#1}
3891                              },
3892      level-1 .initial:n  = { label=\arabic*., nosep, columns=2, first=\small, font=\small },
3893      level-2 .code:n      = \tl_put_right:Nn \l__enumext_print_keyans_ii_tl
3894                              {
3895                                \setenumext[level,2] {#1} \setenumext[print,2] {#1}
3896                              },
3897      level-2 .initial:n  = { nosep, label=(\alph*), first=\small, font=\small },
3898      level-3 .code:n      = \tl_put_right:Nn \l__enumext_print_keyans_iii_tl
3899                              {
3900                                \setenumext[level,3] {#1} \setenumext[print,3] {#1}
3901                              },
3902      level-3 .initial:n  = { nosep, label=\roman*., first=\small, font=\small },
3903      level-4 .code:n      = \tl_put_right:Nn \l__enumext_print_keyans_iv_tl
3904                              {
3905                                \setenumext[level,4] {#1} \setenumext[print,4] {#1}
3906                              },
3907      level-4 .initial:n  = { nosep, label=\Alph*., first=\small, font=\small },
3908      level-* .code:n      = \tl_put_right:Nn \l__enumext_print_keyans_vii_tl % starred
3909                              {
3910                                \setenumext[enumext*] {#1} %%\setenumext[print,*] {#1}
3911                              },
3912      level-* .initial:n  = { label=\arabic*., nosep, columns=2, first=\small, font=\small },
3913    }
```

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

```
3914  \NewDocumentCommand \printkeyans { s O{} m }
3915    {
3916      \group_begin:
3917        \tl_use:N \l__enumext_print_keyans_i_tl
3918        \tl_use:N \l__enumext_print_keyans_ii_tl
3919        \tl_use:N \l__enumext_print_keyans_iii_tl
3920        \tl_use:N \l__enumext_print_keyans_iv_tl
3921        \tl_use:N \l__enumext_print_keyans_vii_tl
3922        \__enumext_printkeyans:nnn { #1 } { #2 } { #3 }
3923      \group_end:
3924    }
```

(*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

```
3925  \cs_new_protected:Npn \__enumext_printkeyans:nnn #1 #2 #3
3926    {
3927      \seq_if_exist:cTF { g__enumext_#3_seq }
3928        {
3929          \seq_if_empty:cF { g__enumext_#3_seq }
3930            {
3931              %%\seq_show:c { g__enumext_#3_seq }
3932              \bool_if:nTF {#1}
3933                {
3934                  \begin{enumext*}[#2]
3935                    \seq_map_inline:cn { g__enumext_#3_seq } { ##1 }
3936                  \end{enumext*}
3937                }
3938                {
3939                  \begin{enumext}[#2]
3940                    \seq_map_inline:cn { g__enumext_#3_seq } { ##1 }
3941                  \end{enumext}
3942                }
3943            }
3944        }
3945        {
3946          \msg_error:nnn { enumext } { undefined-storage-anskey } {#3}
3947        }
3948    }
```

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

### 10.39    The command `\setenumext`

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

```
3949  \keys_define:nn { enumext / meta-families }
3950    {
3951      level-1  .code:n = { \keys_set:nn { enumext / level-1  } {#1} } ,
3952      level-2  .code:n = { \keys_set:nn { enumext / level-2  } {#1} } ,
3953      level-3  .code:n = { \keys_set:nn { enumext / level-3  } {#1} } ,
3954      level-4  .code:n = { \keys_set:nn { enumext / level-4  } {#1} } ,
3955      keyans   .code:n = { \keys_set:nn { enumext / keyans   } {#1} } ,
3956      enumext* .code:n = { \keys_set:nn { enumext / enumext* } {#1} } ,
3957      keyans*  .code:n = { \keys_set:nn { enumext / keyans*  } {#1} } ,
3958      print-1  .code:n = { \keys_set:nn { keyanskey / print } { level-1 = {#1} } } ,
3959      print-2  .code:n = { \keys_set:nn { keyanskey / print } { level-2 = {#1} } } ,
3960      print-3  .code:n = { \keys_set:nn { keyanskey / print } { level-3 = {#1} } } ,
3961      print-4  .code:n = { \keys_set:nn { keyanskey / print } { level-4 = {#1} } } ,
3962      print-*  .code:n = { \keys_set:nn { keyanskey / print } { level-* = {#1} } } ,
3963      unknown  .code:n = { \msg_error:nn { enumext } { unknown-key-family } } ,
3964    }
```

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

```
3965  \seq_const_from_clist:Nn \c__enumext_all_families_seq
3966    {
3967      level-1 , level-2 , level-3 , level-4 , keyans, enumext*,
3968      keyans* , print-1 , print-2 , print-3 , print-4 , print-*,
3969    }
```

`\setenumext`

Now we define the user command `\setenumext`.

```
3970  \NewDocumentCommand \setenumext { o +m }
3971    {
3972      \tl_if_novalue:nTF {#1}
3973        {
3974          \seq_map_inline:Nn \c__enumext_all_families_seq
3975        }
3976        {
3977          \seq_clear:N \l__enumext_setkey_tmpa_seq
3978          \seq_set_from_clist:Nn \l__enumext_setkey_tmpb_seq {#1}
3979          \int_set:Nn \l__enumext_setkey_tmpa_int
```

```
3980                 {
3981                     \seq_count:N \l__enumext_setkey_tmpb_seq
3982                 }
3983             \int_compare:nNnTF { \l__enumext_setkey_tmpa_int } > { 1 }
3984                 {
3985                     \seq_pop_left:NN \l__enumext_setkey_tmpb_seq \l__enumext_setkey_tmpa_tl
3986                     \seq_map_function:NN \l__enumext_setkey_tmpb_seq \__enumext_set_parse:n
3987                     \seq_set_map_e:NNn \l__enumext_setkey_tmpa_seq \l__enumext_setkey_tmpa_seq
3988                         {
3989                             \tl_use:N \l__enumext_setkey_tmpa_tl - ##1
3990                         }
3991                 }
3992                 {
3993                     \seq_put_right:Ne \l__enumext_setkey_tmpa_seq { \tl_trim_spaces:n {#1} }
3994                 }
3995             \seq_if_empty:NTF \l__enumext_setkey_tmpa_seq
3996                 { \seq_map_inline:Nn \c__enumext_all_families_seq }
3997                 { \seq_map_inline:Nn \l__enumext_setkey_tmpa_seq }
3998         }
3999         {
4000             \keys_set:nn { enumext / meta-families } { ##1 = {#2} }
4001         }
4002     }
```

(*End of definition for* \setenumext. *This function is documented on page 5*.)

\__enumext_set_parse:n
\__enumext_set_error:nn

Internal functions used by the \setenumext command.

```
4003 \cs_new_protected:Npn \__enumext_set_parse:n #1
4004     {
4005         \tl_set:Ne \l__enumext_setkey_tmpb_tl { \tl_trim_spaces:n {#1} }
4006         \int_step_inline:nnn { 0 } { 4 } % <- max level
4007             { \tl_remove_all:Nn \l__enumext_setkey_tmpb_tl {##1} }
4008         \tl_if_empty:NTF \l__enumext_setkey_tmpb_tl
4009             {
4010                 \seq_put_right:Ne \l__enumext_setkey_tmpa_seq
4011                     { \tl_trim_spaces:n {#1} }
4012             }
4013             { \__enumext_set_error:nn {#1} { } }
4014     }
4015 \cs_new_protected:Npn \__enumext_set_error:nn #1 #2
4016     { \msg_error:nnn { enumext } { invalid-key } {#1} {#2} }
```

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

## 10.40   Messages

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

```
4017 \msg_new:nnn { enumext } { package-load }
4018     {
4019         The ~ '#1' ~ package ~ is ~ already ~ loaded.
4020     }
4021 \msg_new:nnn { enumext } { package-not-load }
4022     {
4023         The ~ '#1' ~ package ~ will ~ be ~ loaded ~ as ~ a ~ dependency.
4024     }
4025 \msg_new:nnn { enumext } { package-load-foot }
4026     {
4027         The ~ '#1' ~ package ~ is ~ loaded ~ with ~ the ~ option ~ '#2'.
4028     }
```

Message used in the creation of counters by enumext package.

```
4029 \msg_new:nnn { enumext } { counters }
4030     {
4031         The ~ counter ~ '#1' ~ is ~ already ~ defined ~ by ~ some ~ \\
4032         package ~ or ~ macro, ~ it ~ cannot ~ be ~ continued.
4033     }
```

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

```
4034 \msg_new:nnn { enumext } { invalid-key }
4035     {
4036         The ~ key ~ '#1' ~ is ~ not ~ know ~ the ~ level ~ #2.
```

```
4037      }
4038  \msg_new:nnn { enumext } { unknown-key-family }
4039      {
4040        Unknown~key~family~`\l_keys_key_str'~for~enumext.
4041      }
```

Messages used in length calculation.

```
4042  \msg_new:nnn { enumext } { width-negative }
4043      {
4044        Ignoring ~ negative ~ value ~ '#1=#2' ~ \msg_line_context:.\\
4045        The ~ key ~ '#1'~ accepts ~ values  ~ >= ~ 0pt.
4046      }
4047  \msg_new:nnn { enumext } { width-zero }
4048      {
4049        Invalid ~ '#1=#2' ~ \msg_line_context:.\\
4050        The ~ key ~ '#1'~ accepts ~ values  ~ > ~ 0pt.
4051      }
```

Messages used by show-length key in enumext.

```
4052  \msg_new:nnn { enumext } { list-lengths }
4053      {
4054        **** ~ Lengths ~ used ~ by ~ 'enumext' ~ level ~ '#2' ~ \msg_line_context:~\c_space_tl ****\\
4055        \__enumext_show_length:nnn { dim  } { labelsep     } {#1}
4056        \__enumext_show_length:nnn { dim  } { labelwidth   } {#1}
4057        \__enumext_show_length:nnn { dim  } { itemindent   } {#1}
4058        \__enumext_show_length:nnn { dim  } { leftmargin   } {#1}
4059        \__enumext_show_length:nnn { dim  } { rightmargin  } {#1}
4060        \__enumext_show_length:nnn { dim  } { listparindent } {#1}
4061        \__enumext_show_length:nnn { skip } { topsep     } {#1}
4062        \__enumext_show_length:nnn { skip } { parsep     } {#1}
4063        \__enumext_show_length:nnn { skip } { partopsep } {#1}
4064        \__enumext_show_length:nnn { skip } { itemsep    } {#1}
4065        ****************************************************
4066      }
```

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

```
4067  \msg_new:nnn { enumext } { list-lengths-not-nested }
4068      {
4069        **** ~ Lengths ~ used ~ by ~ '#2' ~ environment ~ \msg_line_context:~\c_space_tl ****\\
4070        \__enumext_show_length:nnn { dim  } { labelsep     } {#1}
4071        \__enumext_show_length:nnn { dim  } { labelwidth   } {#1}
4072        \__enumext_show_length:nnn { dim  } { itemindent   } {#1}
4073        \__enumext_show_length:nnn { dim  } { leftmargin   } {#1}
4074        \__enumext_show_length:nnn { dim  } { rightmargin  } {#1}
4075        \__enumext_show_length:nnn { dim  } { listparindent } {#1}
4076        \__enumext_show_length:nnn { skip } { topsep     } {#1}
4077        \__enumext_show_length:nnn { skip } { parsep     } {#1}
4078        \__enumext_show_length:nnn { skip } { partopsep } {#1}
4079        \__enumext_show_length:nnn { skip } { itemsep    } {#1}
4080        ****************************************************
4081      }
```

Messages used by ref key.

```
4082  \msg_new:nnn { enumext } { key-ref-empty }
4083      {
4084        Key ~ 'ref' ~ need ~ a ~ value ~ in ~ '#1'~ \msg_line_context:.
4085      }
```

Messages used by save-ans key.

```
4086  \msg_new:nnn { enumext } { save-ans-empty }
4087      {
4088        Key ~ 'save-ans' ~ need ~ a ~ value ~ in ~ '#1'~ \msg_line_context:.
4089      }
4090
4091  \msg_new:nnn { enumext } { save-ans-ok }
4092      {
4093        Set ~ 'save-ans=#2' ~ in ~'#1' ~ \msg_line_context:.
4094      }
```

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

```
4095  \msg_new:nnn { enumext } { items-same-answer }
4096      {
4097        **********~Checking~answers~on~'#1'~OK~**********\\
4098        **~ All ~ items ~ stored ~ in ~ sequence ~ '#1' ~ have ~ an ~ answer. \\
```

```
4099        *****************************************
4100        \prg_replicate:nn { 7 + \str_count:n {#1} } { * }
4101    }
4102 \msg_new:nnn { enumext } { item-different-answer }
4103    {
4104      Number ~ of ~ items ~ different ~  of ~ number ~ of ~
4105      answer ~ in ~ sequence ~ '#1'~ closed ~ \msg_line_context:.
4106    }
```

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

```
4107 \msg_new:nnn { enumext } { missing-starred }
4108    {
4109      Missing ~ '\c_backslash_str #1*' ~ in ~ '#2' ~ \msg_line_context:.
4110    }
```

Message for the nesting depth of the environment enumext.

```
4111 \msg_new:nnn { enumext } { list-too-deep }
4112    {
4113      Too ~ deep ~ nesting  ~ for ~ 'enumext' ~ \msg_line_context:.~ \\
4114      The ~ maximum  ~ level  ~ of  ~ nesting ~ is ~ 4.
4115    }
```

Messages used by \anskey and \anspic commands.

```
4116 \msg_new:nnn { enumext } { anskey-wrong-place }
4117    {
4118      Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:.~ \\
4119      '\c_backslash_str #1' ~ works ~ in ~ the ~ environment ~ '#2'.
4120    }
4121 \msg_new:nnn { enumext } { anspic-wrong-place }
4122    {
4123      Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:.~ \\
4124      '\c_backslash_str #1' ~ works ~ in ~ the ~ environment ~ '#2'.
4125    }
4126 \msg_new:nnn { enumext } { command-wrong-place }
4127    {
4128      Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:.~ \\
4129      '\c_backslash_str #1' ~ works ~ outside ~ the ~ environment ~ '#2'.
4130    }
```

Messages used by keyans and keyanspic environment.

```
4131 \msg_new:nnn { enumext } { keyans-nested }
4132    {
4133      The ~ environment ~ 'keyans' ~ can't ~ be  ~ nested  ~ \msg_line_context:.
4134    }
4135 \msg_new:nnn { enumext } { keyans-wrong-level }
4136    {
4137      Wrong ~ level ~ position ~ for ~ 'keyans' ~ \msg_line_context:.~ \\
4138      The ~ environment ~ 'keyans' ~ can ~ only ~ be ~ in ~ the ~ first ~ level.
4139    }
4140 \msg_new:nnn { enumext } { wrong-place }
4141    {
4142      Wrong ~ place ~ for ~ '#1' ~ environment ~\msg_line_context:.~ \\
4143      '#1' ~ is ~ only ~ found ~ with ~ '#2' ~  in  ~  'enumext.
4144    }
4145 \msg_new:nnn { enumext } { keyanspic-nested }
4146    {
4147      The ~ environment ~ 'keyanspic' ~ can't ~ be  ~ nested~ \msg_line_context:.~.
4148    }
4149 \msg_new:nnn { enumext } { keyanspic-wrong-level }
4150    {
4151      Wrong ~ level ~ position ~ for ~ 'keyanspic' ~ \msg_line_context:.~ \\
4152      The ~ environment ~ 'keyans' ~ can ~ only ~ be ~ in ~ the ~ first ~ level.
4153    }
```

Messages used by \getkeyans command.

```
4154 \msg_new:nnn { enumext } { undefined-storage-anskey }
4155    {
4156      Storage ~ named ~ '#1' ~ is ~ not ~ defined ~ \msg_line_context:.
4157    }
```

Messages used by \miniright command.

```
4158 \msg_new:nnn { enumext } { missing-miniright }
4159    {
4160      Missing ~ '\c_backslash_str miniright' ~ in ~ \msg_line_context:.\\
```

```
4161        The ~ key ~ 'mini-env' ~ need ~ '\c_backslash_str miniright'.
4162    }
4163 \msg_new:nnn { enumext } { wrong-miniright-place }
4164    {
4165      Wrong ~ place ~ for ~ '\c_backslash_str miniright' ~ \msg_line_context:.~ \\
4166      Works ~ in ~ 'enumext' ~ and ~ 'keyans' ~ with ~ key ~ 'mini-env'.
4167    }
4168 \msg_new:nnn { enumext } { wrong-miniright-use }
4169    {
4170      Wrong ~ use ~ for ~ '\c_backslash_str miniright' ~ \msg_line_context:.~ \\
4171      '\c_backslash_str miniright' ~ need ~ a ~ key ~ 'mini-env'.
4172    }
```

Messages used by enumext* and keyans* environments.

```
4173 \msg_new:nnn { enumext } { nested }
4174    {
4175      The ~ starred ~ environment ~ can't ~ be ~ nested ~ \msg_line_context:.
4176    }
4177 \msg_new:nnn { enumext } { item-joined }
4178    {
4179      Items ~ joined ~ (#1) ~ > ~ #2  ~ columns ~\msg_line_context:.
4180    }
4181 \msg_new:nnn { enumext } { item-joined-columns }
4182    {
4183      Not ~ space ~ to ~ join ~ items ~ (#1) ~ > ~ #2 ~\msg_line_context:.
4184    }
```

## 10.41    Finish package

Finish package implementation.

```
4185 \file_input_stop:
4186 ⟨/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.