enumext

ENUMERATE EXERCISE SHEETS

V1.0   2024-05-06*

©2024 by Pablo González†

Abstract

This package provides *"enumerated list"* environments for creating *"simple exercise sheets"* along with *"multiple choice questions"*, storing the ⟨*answers*⟩ to these in memory using the `multicol` package and the `l3seq` and `l3prop` modules.

Contents

Motivation and acknowledgments

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

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

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

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

License and Requirements

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.

---

*This file describes a documentation for v1.0, last revised 2024-05-06.
†E-mail: «pablgonz@educarchile.cl».

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

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

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.
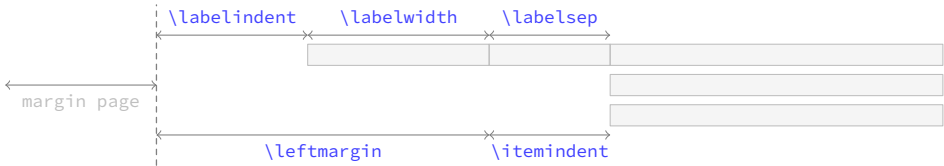

Figure 1: Representation of horizontal lengths in enumitem.

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.
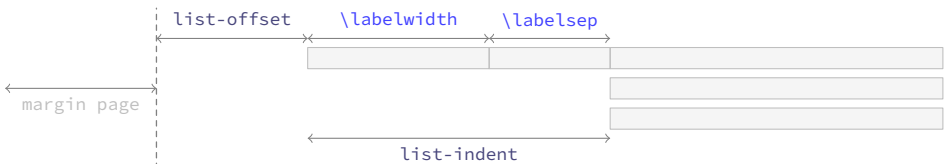
Figure 2: Representation of horizontal lengths concept in enumext.

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.

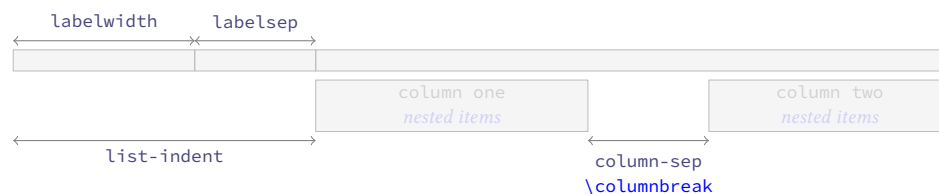The *"non starred"* version of the `multicols` environment is always used together with the `\raggedcolumns` command and is controlled by `columns` and `columns-sep` keys. The environment is available for all nesting levels, and can can together with the `mini-env` key. If you need to force a start a 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 `store-ref` (§4.2) when the key `save-ans`(§4.1) is active.

🎯 The implementation of `\label` and `\ref` together with the `store-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 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 `\dimexpr` or `\dimeval` 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 and the form in which it is used by standard *"label and ref"* and the *"internal reference"* system with the `store-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 `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*⟩} 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*⟩`}}`. The {⟨*code*⟩} must be passed between braces.

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

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

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 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.

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

Sets the *start* to value from the previous of the counter defined by label key for the *"first level"*. This ⟨*key*⟩ does not receive an argument. The ⟨*key*⟩ can be overwritten using the start key. If the save-ans key is present and {⟨*store name*⟩} exist, the numbering will continue according to this key. This key is *"only"* available for the *"first level"* of enumext.

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.

4 The storage system

The entire mechanism for *"storing content"* it is activated according to save-ans key on the *"first level"* of enumext environment. Only when this ⟨*key*⟩ is *"active"* the \anskey command and the environments 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` environment, `\item*` in `keyans` environment and `\anspic*` in `keyanspic` environment. If the ⟨*sequence*⟩ or ⟨*prop list*⟩ does not exist, it will be created globally.

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

Wraps the current ⟨*argument*⟩ passed `\anskey` command to referenced by {#1}. The {⟨*code*⟩} must be passed between braces. This ⟨*key*⟩ 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}'.

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.

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 `store-ref` key is active.

4.2 Keys for internal `label` and `ref`

store-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 ⟨*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 `store-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 check answers

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 `store-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]
  \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]
  \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}
```
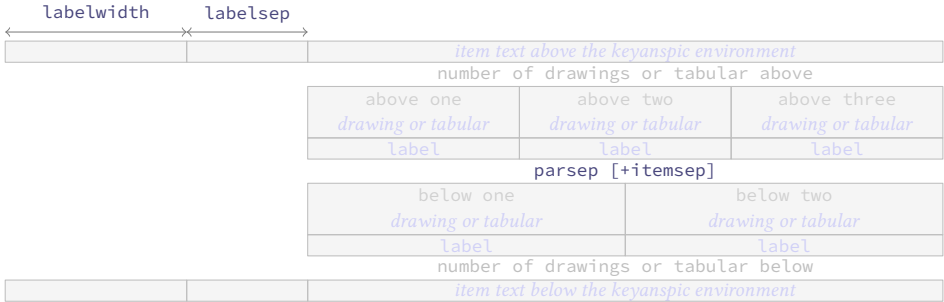
1. Question with images.

(A)

(B)

(C)

(D)

* (E) [note]
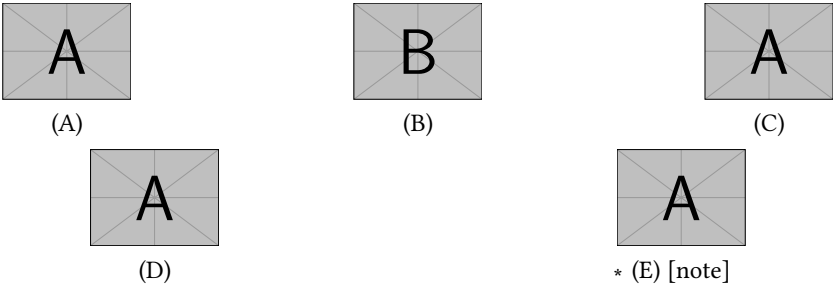
4.7 Printing stored content

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 `store-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,nosep,store-ref]
  \item Factor $3x+3y+3z$. \anskey{$3(x+y+z)$}
  \item True False

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

  \item Related to Linux

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

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

\printkeyans{sample}
```

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

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

2. True False

   (a) LaTeX2e is cool?

[2] Very True!

3. Related to Linux

   (a) You use linux?

[3] Yes

   (b) Rate the following package and class

     i. `xsim`

[4] very good

     ii. `exsheets`

[5] obsolete

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

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

5 Full examples

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

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

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

Example 1

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

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

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

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

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

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

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

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

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

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

Example 2

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

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

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

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

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

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

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

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

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

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

---

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

Example 3

A *"simple multiple choice"* test 📄.

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

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

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

4. Question with image and label below:

   Ⓐ

   Ⓑ

   Ⓒ

   Ⓓ

   Ⓔ

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

Test keys

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

Example 4

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

🦆 1. Factor $x^2 - 2x + 1$

🦆 2. Factor $3x + 3y + 3z$

The following questions need to be cuaqtified :)

🦆 3. True False
   - (a) $\alpha > \delta$
   - (b) $\LaTeX$2e is cool?

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

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

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

Example 5

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

1

Which choice best describes what happens in the passage?

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

2

Which choice best describes what happens in the passage?

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

3

Which choice best describes what happens in the passage?

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

4

Which choice best describes what happens in the passage?

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

1. A)　　　2. C)　　　3. B)　　　4. D)

6　The way of non-enumerated lists

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

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

Fake `itemize` environment

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

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

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

Fake `description` environment

Here we set `label={}` and `list-indent=2.5`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 2_ε: An Unofficial Reference Manual". Available from ctan, `https://ctan.org/pkg/latex2e-help-texinfo`, 2024.

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

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

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

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

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

[12] Gundlach, Patrick. "The lua-visual-debug package". Available from ctan, `https://www.ctan.org/pkg/lua-visual-debug`, 2023.

[13] Lemvig, Mogens. "The shortlst package". Available from ctan, `https://www.ctan.org/pkg/shortlst`, 1998.

[14] Niederberger, Clemens. "tasks – Horizontally columned lists". Available from ctan, `https://www.ctan.org/pkg/tasks`, 2022.

8   Change history

v1.0 2024-05-06   – 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    {2024-05-06}
7    {1.0}
8    {Enumerate exercise sheets}
```

Finally check if the multicol package is loaded, if not we load it.

```
9   \hook_gput_code:nnn {begindocument} {enumext}
10    {
11      \IfPackageLoadedTF { multicol }
12        {
13          \msg_info:nnn { enumext } { package-load } { multicol }
14        }
15        {
16          \msg_info:nnn { enumext } { package-not-load } { multicol }
17          \RequirePackage{multicol}[2023-03-30]
18        }
19    }
```

10.4   Definition of variables

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

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

```
\l__enumext_level_int
\l__enumext_level_h_int
\l__enumext_keyans_level_int
\l__enumext_keyans_level_h_int
\l__enumext_keyans_pic_level_int
\l__enumext_starred_bool
\g__enumext_starred_bool
\l__enumext_standar_bool
\g__enumext_standar_bool
\l__enumext_keyans_env_bool
```

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

```
27  \bool_new:N \l__enumext_standar_bool
28  \bool_new:N \g__enumext_standar_bool
29  \bool_new:N \l__enumext_keyans_env_bool
```

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

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

Variables to store the *"name of the counters"* enumXi, enumXii, enumXiii and enumXiv for enumext environment, enumXv for keyans environment and enumXvi for the keyanspic environment. The counters enumXvii and enumXviii are used by enumext* and keyans* environments. The initial values of these variables are set by the function __enumext_define_counters:Nn and then modified by the function __enumext_label_style:Nnn used by label key (§10.8).

```
30  \cs_set_protected:Npn \__enumext_tmp:n #1
31    {
32      \tl_new:c { l__enumext_counter_#1_tl }
33    }
34  \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.*)

\l__enumext_resume_bool
\g__enumext_resume_int
\l__enumext_resume_vii_bool
\g__enumext_resume_vii_int
\g__enumext_item_symbol_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 (§10.21). The global token list \g__enumext_item_symbol_tl is used by item-sym* key (§10.26).

```
35  \bool_new:N \l__enumext_resume_bool
36  \int_new:N  \g__enumext_resume_int
37  \bool_new:N \l__enumext_resume_vii_bool
38  \int_new:N  \g__enumext_resume_vii_int
39  \tl_new:N   \g__enumext_item_symbol_tl
```

(*End of definition for* \l__enumext_resume_bool *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.

```
40  \dim_new:N \l__enumext_current_widest_dim
41  \tl_new:N  \g__enumext_counter_styles_tl
42  \tl_new:N  \g__enumext_widest_label_tl
43  \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 (§10.30) which determines the internal values for \leftmargin and \itemindent.

```
44  \cs_set_protected:Npn \__enumext_tmp:n #1
45    {
46      \bool_new:c { l__enumext_leftmargin_tmp_#1_bool }
47      \dim_new:c  { l__enumext_leftmargin_tmp_#1_dim }
48      \dim_new:c  { l__enumext_leftmargin_#1_dim     }
49      \dim_new:c  { l__enumext_itemindent_#1_dim     }
50    }
51  \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).

```
52  \cs_set_protected:Npn \__enumext_tmp:n #1
53    {
54      \skip_new:c  { l__enumext_multicols_above_#1_skip }
55      \skip_new:c  { l__enumext_multicols_below_#1_skip }
56    }
57  \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.)

The left margin lists:

`\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).

```
58  \int_new:N  \g__enumext_minipage_stat_int
59  \skip_new:N \l__enumext_minipage_left_skip
60  \skip_new:N \l__enumext_minipage_right_skip
61  \skip_new:N \l__enumext_minipage_after_skip
62  \skip_new:N \g__enumext_minipage_right_skip
63  \skip_new:N \g__enumext_minipage_after_skip
64  \cs_set_protected:Npn \__enumext_tmp:n #1
65    {
66      \dim_new:c  { l__enumext_minipage_left_#1_dim    }
67      \bool_new:c { l__enumext_minipage_active_#1_bool }
68    }
69  \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.*)

The left margin lists:

`\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

```
70  \cs_set_protected:Npn \__enumext_tmp:n #1
71    {
72      \bool_new:c { l__enumext_wrap_label_#1_bool     }
73      \bool_new:c { l__enumext_wrap_label_opt_#1_bool }
74      \int_new:c  { l__enumext_start_#1_int            }
75      \tl_new:c   { l__enumext_fake_item_indent_#1_tl }
76      \tl_new:c   { l__enumext_label_fill_left_#1_tl  }
77      \tl_new:c   { l__enumext_label_fill_right_#1_tl }
78      \bool_new:c { l__enumext_vspace_a_star_#1_bool  }
79      \bool_new:c { l__enumext_vspace_b_star_#1_bool  }
80    }
81  \clist_map_inline:nn { i, ii, iii, iv, v, vii, viii } { \__enumext_tmp:n {#1} }
```

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

The left margin lists:

`\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_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 (§10.21) 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 (§10.21).

The variable `\l__enumext_store_anskey_arg_tl` stores the contents of `\anskey` (§10.24) and the variable `\l__enumext_store_keyans_label_tl` stores the contents of `\item*` (§10.28.2) 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.

```
82  \bool_new:N \l__enumext_store_active_bool
83  \tl_new:N   \l__enumext_store_name_tl
84  \tl_new:N   \g__enumext_store_name_tl
85  \tl_new:N   \l__enumext_store_anskey_arg_tl
86  \int_new:N  \l__enumext_store_columns_join_int
87  \tl_new:N   \l__enumext_store_keyans_label_tl
88  \tl_new:N   \l__enumext_keyans_tmpa_tl
89  \tl_new:N   \l__enumext_keyans_tmpb_tl
90  \dim_new:N  \l__enumext_keyans_tmpa_dim
```

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

The left margin lists:

`\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).

```
91  \tl_new:N  \l__enumext_setkey_tmpa_tl
92  \tl_new:N  \l__enumext_setkey_tmpb_tl
93  \int_new:N \l__enumext_setkey_tmpa_int
94  \seq_new:N \l__enumext_setkey_tmpa_seq
95  \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*.

```
96  \cs_set_protected:Npn \__enumext_tmp:n #1
97    {
98      \tl_new:c { l__enumext_store_opt_#1_tl            }
99      \tl_new:c { l__enumext_print_keyans_#1_tl         }
100     \bool_new:c { l__enumext_store_columns_#1_bool     }
101     \int_new:c { l__enumext_store_columns_#1_int       }
102     \bool_new:c { l__enumext_store_columns_sep_#1_bool }
103     \dim_new:c { l__enumext_store_columns_sep_#1_dim   }
104     \bool_new:c { l__enumext_store_upper_level_#1_bool }
105   }
106 \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 (§10.24), keyans and keyanspic environments. These variables are used by show-ans, show-pos, mark-ans, save-key and mark-ref keys (§10.23).

```
107 \bool_new:N \l__enumext_show_answer_bool
108 \bool_new:N \l__enumext_show_position_bool
109 \tl_new:N   \l__enumext_mark_ref_sym_tl
110 \tl_new:N   \l__enumext_mark_answer_sym_tl
111 \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).

```
112 \seq_new:N  \l__enumext_keyans_pic_body_seq
113 \dim_new:N  \l__enumext_keyans_pic_width_dim
114 \int_new:N  \l__enumext_keyans_pic_above_int
115 \int_new:N  \l__enumext_keyans_pic_below_int
116 \skip_new:N \l__enumext_keyans_pic_above_skip
```

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

`\l__enumext_store_ans_bool`
`\l__enumext_check_ans_bool`
`\g__enumext_check_ans_show_bool`
`\g__enumext_check_ans_show_h_bool`
`\g__enumext_check_ans_item_tl`
`\g__enumext_count_item_anskey_int`
`\g__enumext_count_item_number_int`

Internal variables used by *"check answer"* mechanism (§10.22.1) controlled by the check-ans and no-store keys.

```
117 \bool_new:N \l__enumext_store_ans_bool
118 \bool_new:N \l__enumext_check_ans_bool
119 \bool_new:N \g__enumext_check_ans_show_bool
120 \bool_new:N \g__enumext_check_ans_show_h_bool
121 \tl_new:N   \g__enumext_check_ans_item_tl
122 \int_new:N  \g__enumext_count_item_anskey_int
123 \int_new:N  \g__enumext_count_item_number_int
124 \int_new:N  \g__enumext_standar_star_env_int
125 \int_new:N  \g__enumext_starred_star_env_int
126 \int_new:N  \g__enumext_starred_keyans_star_env_int
127 \int_new:N  \g__enumext_standar_keyans_star_env_int
128 \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.

```
129 \bool_new:N \l__enumext_hyperref_bool
130 \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 store-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.

```
131 \tl_new:N \l__enumext_newlabel_arg_one_tl
132 \tl_new:N \l__enumext_newlabel_arg_two_tl
```

```
133 \tl_new:N \l__enumext_store_write_aux_file_tl
134 \cs_set_protected:Npn \__enumext_tmp:n #1
135   {
136     \tl_new:c { l__enumext_label_copy_#1_tl }
137   }
138 \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.

```
139 \int_new:N \g__enumext_footnote_int
140 \seq_new:N \g__enumext_footnote_arg_seq
141 \seq_new:N \g__enumext_footnote_int_seq
```

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

\c__enumext_counter_style_tl
\l__enumext_ref_key_arg_tl
\l__enumext_ref_aux_tl
\l__enumext_the_counter_X_tl
\l__enumext_counter_style_for_ref_X_tl

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

```
142 \tl_const:Nn \c__enumext_counter_style_tl
143   { { arabic } { roman } { Roman } { alph } { Alph } }
144 \tl_new:N \l__enumext_ref_key_arg_tl
145 \tl_new:N \l__enumext_ref_aux_tl
146 \cs_set_protected:Npn \__enumext_tmp:n #1
147   {
148     \tl_new:c  { l__enumext_counter_style_for_ref_#1_tl }
149     \tl_new:c  { l__enumext_the_counter_#1_tl }
150     \tl_set:ce { l__enumext_the_counter_#1_tl } { \exp_not:c { theenumX#1 } }
151   }
152 \clist_map_inline:nn { i, ii, iii, iv, v, vi, vii, viii } { \__enumext_tmp:n {#1} }
```

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

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

Internal variables used by enumext* and keyans* environments.

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

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

(*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.

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

(*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.

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

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

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

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

(*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?.

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

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

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

```
196  \cs_new:Npn \__enumext_show_length:nnn #1 #2 #3
197    {
198      * ~ #2
199      \prg_replicate:nn { 14 - \str_count:n {#2} } { ~ }
200        = ~ \use:c { #1_use:c } { l__enumext_#2_#3_#1 } \\
201    }
```

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

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

```
202  \cs_set_protected:Nn \__enumext_zero_count_level:
203    {
204      \cs_set_protected:Npn \__enumext_tmp:n ##1
205        {
206          \int_gzero:c { g__enumext_count_level_##1_int }
207        }
208      \clist_map_inline:nn { i, ii, iii, iv, vii } { \__enumext_tmp:n {##1} }
209    }
```

(*End of definition for* __enumext_zero_count_level:.)

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.

```
210 \__enumext_at_begin_document:n
211   {
212     \cs_new_eq:NN \__enumext_start_list:nn \list
213     \cs_new_eq:NN \__enumext_stop_list: \endlist
214     \cs_new_eq:NN \__enumext_item_std:w \item
215   }
```

(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.

```
216 \__enumext_at_begin_document:n
217   {
218     \cs_new_eq:NN \__enumext_minipage:w \minipage
219     \cs_new_eq:NN \__enumext_endminipage: \endminipage
220   }
```

(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.

```
221 \hook_gput_code:nnn { begindocument } { enumext } { \__enumext_after_hyperref: }
222 \hook_gset_rule:nnnn { begindocument } { enumext } { after } { hyperref }
```

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

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

```
223 \cs_new_protected:Nn \__enumext_after_hyperref:
224   {
225     \IfPackageLoadedTF { hyperref }
226       {
227         \msg_info:nnn { enumext } { package-load } { hyperref }
228         \bool_set_true:N \l__enumext_hyperref_bool
229         \IfHyperBoolean{hyperfootnotes}
230           {
231             \typeout{hyperfootnotes=true}
232             \bool_set_true:N \l__enumext_footnotes_key_bool
233           }
234           { \typeout{hyperfootnotes=false} }
235       }
236       {  }
```

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`.

```
237     \bool_if:NT \l__enumext_footnotes_key_bool
238       {
239         \IfPackageLoadedTF { footnotehyper }
240           {
```

```
241        \msg_info:nnn { enumext } { package-load } { footnotehyper }
242      }
243      {
244        \typeout{No ~ footnotehyper ~ load}
245        \typeout{Load ~ and  ~ use  ~ \string\makesavenoteenv{enumext*}}
246        \bool_set_false:N \l__enumext_footnotes_key_bool
247      }
248    }
```

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.

```
249    \bool_if:NTF \l__enumext_hyperref_bool
250      {
251        \cs_new_eq:NN \__enumext_hypertarget:nn \hypertarget
252        \cs_new_eq:NN \__enumext_phantomsection: \phantomsection
253      }
254      {
255        \cs_new_eq:NN \__enumext_hypertarget:nn \use_none:nn
256        \cs_new_eq:NN \__enumext_phantomsection: \prg_do_nothing:
257      }
258    }
```

(*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 store-
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.

```
259  \cs_new_protected:Npn \__enumext_newlabel:nn #1 #2
260    {
261      \protected@write \@auxout { }
262        {
263          \token_to_str:N \newlabel {#1}
264            {
265              {#2}
266              \bool_if:NT \l__enumext_hyperref_bool
267                { { \thepage } {#2} {#1} }
268              { }
269            }
270        }
271      \__enumext_hypertarget:nn {#1} { }
272      \__enumext_phantomsection:
273    }
```

(*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.

```
274  \cs_new_protected:Npn \__enumext_define_counters:Nn #1 #2
275    {
276      \cs_if_exist:cTF { c@ #2 }
277        { \msg_fatal:nnn { enumext } { counters }{ #2 } }
278        {
279          \tl_set:Nn #1 { #2 }
280          \newcounter { #2 }
281        }
282    }
```

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

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.

enumXi
enumXii
enumXiii
enumXiv
enumXv
enumXvi
enumXvii
enumXviii

```
283 \__enumext_define_counters:Nn \l__enumext_counter_i_tl    { enumXi     }
284 \__enumext_define_counters:Nn \l__enumext_counter_ii_tl   { enumXii    }
285 \__enumext_define_counters:Nn \l__enumext_counter_iii_tl  { enumXiii   }
286 \__enumext_define_counters:Nn \l__enumext_counter_iv_tl   { enumXiv    }
287 \__enumext_define_counters:Nn \l__enumext_counter_v_tl    { enumXv     }
288 \__enumext_define_counters:Nn \l__enumext_counter_vi_tl   { enumXvi    }
289 \__enumext_define_counters:Nn \l__enumext_counter_vii_tl  { enumXvii   }
290 \__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.

```
291 \cs_new_protected:Npn \__enumext_register_counter_style:Nn #1 #2
292   {
293     \tl_const:cn { c__enumext_widest_ \cs_to_str:N #1 _tl } {#2}
294     \tl_gput_right:Nn \g__enumext_counter_styles_tl {#1}
295   }
296 \__enumext_register_counter_style:Nn \arabic { 0 }
297 \__enumext_register_counter_style:Nn \Alph   { M }
298 \__enumext_register_counter_style:Nn \alph   { m }
299 \__enumext_register_counter_style:Nn \Roman  { VIII }
300 \__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.

```
301 \cs_new_protected:Npn \__enumext_label_width_by_box:Nn #1 #2
302   {
303     \hbox_set:Nn \l__enumext_label_width_by_box {#2}
304     \dim_set:Nn #1 { \box_wd:N \l__enumext_label_width_by_box }
305   }
306 \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.

```
307 \cs_new_protected:Npn \__enumext_label_style:Nnn #1 #2 #3
308   {
309     \tl_clear_new:N #1
310     \tl_put_right:Ne #1 { \tl_trim_spaces:n {#3} }
311     \tl_gset_eq:NN \g__enumext_widest_label_tl #1
312     \tl_map_inline:Nn \g__enumext_counter_styles_tl
313       {
314         \tl_replace_all:Nne #1 { ##1* } { \exp_not:N ##1 {#2} }
315         \tl_greplace_all:Nne \g__enumext_widest_label_tl { ##1* }
316           { \tl_use:c { c__enumext_widest_ \cs_to_str:N ##1 _tl } }
317       }
318     \__enumext_label_width_by_box:Nn \l__enumext_current_widest_dim
319       { \tl_use:N \g__enumext_widest_label_tl }
320     \tl_set_eq:cN { the #2 } #1
321   }
322 \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.

```
323  \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
324    {
325      \keys_define:nn { enumext / #1 }
326        {
327          font       .tl_set:c   = { l__enumext_label_font_style_#2_tl },
328          font       .value_required:n = true,
329          labelsep   .dim_set:c   = { l__enumext_labelsep_#2_dim },
330          labelsep   .initial:n  = {0.3333em},
331          labelsep   .value_required:n = true,
332          labelwidth .dim_set:c   = { l__enumext_labelwidth_#2_dim },
333          labelwidth .value_required:n = true,
334          wrap-label  .cs_set_protected:cp = { __enumext_wrapper_label_#2:n } ##1,
335          wrap-label  .initial:n  = {##1},
336          wrap-label  .value_required:n = true,
337          wrap-label* .code:n = {
338                                    \bool_set_true:c { l__enumext_wrap_label_opt_#2_bool }
339                                    \keys_set:nn { enumext / #1 } { wrap-label = {##1} }
340                                 },
341          wrap-label* .value_required:n = true,
342        }
343    }
344  \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.

```
345  \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
346    {
347      \keys_define:nn { enumext / #1 }
348        {
349          align .choice:,
350          align / left    .code:n =
351                            {
352                              \tl_clear:c { l__enumext_label_fill_left_#2_tl  }
353                              \tl_set:cn  { l__enumext_label_fill_right_#2_tl } { \hfill }
354                            },
355          align / right   .code:n =
356                            {
357                              \tl_set:cn  { l__enumext_label_fill_left_#2_tl  } { \hfill }
358                              \tl_clear:c { l__enumext_label_fill_right_#2_tl }
359                            },
360          align / center  .code:n =
361                            {
362                              \tl_set:cn { l__enumext_label_fill_left_#2_tl  } { \hfill }
363                              \tl_set:cn { l__enumext_label_fill_right_#2_tl } { \hfill }
364                            },
365          align .initial:n  = left,
366          align .value_required:n  = true,
367        }
368    }
369  \clist_map_inline:nn
370    {
371      {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {keyans}{v}
372    }
373    { \__enumext_tmp:nn #1 }
```

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

```
374  \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
375    {
376      \keys_define:nn { enumext / #1 }
377        {
378          align .choice:,
379          align / left   .code:n = \str_set:cn { l__enumext_align_label_#2_str } { l },
380          align / right  .code:n = \str_set:cn { l__enumext_align_label_#2_str } { r },
```

```
381        align / center .code:n = \str_set:cn { l__enumext_align_label_#2_str } { c },
382        align .initial:n = left,
383        align .value_required:n = true,
384      }
385    }
386 \clist_map_inline:nn { {enumext*}{vii}, {keyans*}{viii} } { \__enumext_tmp:nn #1 }
```

(*End of definition for* align.)

10.11 Setting label and ref keys

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

It loops through the defined counter styles in \c__enumext_counter_style_tl and replace * by real command, for example, looking for \arabic* and replacing that by \arabic{⟨counter⟩} defined on the current level.

```
387 \cs_new_protected:Nn \__enumext_regex_label_ref_key:
388   {
389     \tl_map_inline:Nn \c__enumext_counter_style_tl
390       {
391         \regex_replace_once:nnN { \c{##1}\* }
392           { \c{##1}\cB{\u{l__enumext_ref_aux_tl}\cE} } \l__enumext_ref_key_arg_tl
393       }
394   }
```

(*End of definition for* __enumext_regex_label_ref_key:.)

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

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

```
395 \cs_new_protected:Npn \__enumext_set_label_ref:n #1
396   {
397     \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
398     \tl_set_eq:Nc \l__enumext_ref_aux_tl { l__enumext_counter_ \__enumext_level: _tl }
399     \__enumext_regex_label_ref_key:
400     \tl_set_eq:Nc \l__enumext_ref_aux_tl { l__enumext_the_counter_ \__enumext_level: _tl }
401     \tl_put_right:ce { l__enumext_counter_style_for_ref_ \__enumext_level: _tl }
402       {
403         \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_aux_tl }
404           { \exp_not:V \l__enumext_ref_key_arg_tl }
405       }
406   }
```

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

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

```
407 \cs_new_protected:Nn \__enumext_use_key_ref:
408   {
409     \tl_if_empty:cF { l__enumext_counter_style_for_ref_ \__enumext_level: _tl }
410       {
411         \tl_use:c { l__enumext_counter_style_for_ref_ \__enumext_level: _tl }
412       }
413   }
```

(*End of definition for* __enumext_use_key_ref:.)

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

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

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

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

```
414 \cs_new_protected:Npn \__enumext_set_label_ref_h:n #1
415   {
```

```
416    \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
417    \int_compare:nNnTF { \l__enumext_level_h_int } = { 1 }
418      {
419        \tl_set_eq:NN \l__enumext_ref_aux_tl \l__enumext_counter_vii_tl
420        \__enumext_regex_label_ref_key:
421        \tl_set_eq:NN \l__enumext_ref_aux_tl \l__enumext_the_counter_vii_tl
422        \tl_put_right:Ne \l__enumext_counter_style_for_ref_vii_tl
423          {
424            \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_aux_tl }
425              { \exp_not:V \l__enumext_ref_key_arg_tl }
426          }
427      }
428      {
429        \tl_set_eq:NN \l__enumext_ref_aux_tl \l__enumext_counter_viii_tl
430        \__enumext_regex_label_ref_key:
431        \tl_set_eq:NN \l__enumext_ref_aux_tl \l__enumext_the_counter_viii_tl
432        \tl_put_right:Ne \l__enumext_counter_style_for_ref_vii_tl
433          {
434            \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_aux_tl }
435              { \exp_not:V \l__enumext_ref_key_arg_tl }
436          }
437      }
438    }
```

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

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

```
439  \cs_new_protected:Nn \__enumext_use_key_ref_h:
440    {
441      \int_compare:nNnTF { \l__enumext_level_h_int } = { 1 }
442        {
443          \tl_if_empty:NF \l__enumext_counter_style_for_ref_vii_tl
444            {
445              \tl_use:N \l__enumext_counter_style_for_ref_vii_tl
446            }
447        }
448        {
449          \tl_if_empty:NF \l__enumext_counter_style_for_ref_viii_tl
450            {
451              \tl_use:N \l__enumext_counter_style_for_ref_viii_tl
452            }
453        }
454    }
```

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

10.11.1 Define and set label key for enumext environment

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

`\l__enumext_label_i_tl`
`\l__enumext_label_ii_tl`
`\l__enumext_label_iii_tl`
`\l__enumext_label_iv_tl`

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

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

10.11.2 Define and set `label` key for `enumext*` and `keyans*` environments

label

ref

\l__enumext_label_vii_tl

\l__enumext_label_viii_tl

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

```
475 \cs_set_protected:Npn \__enumext_tmp:nnn #1 #2 #3
476   {
477     \keys_define:nn { enumext / #1 }
478       {
479         label .code:n    = {
480                           \__enumext_label_style:cvn { l__enumext_label_#2_tl }
481                             { l__enumext_counter_#2_tl } {##1}
482                           \dim_set_eq:cN  { l__enumext_labelwidth_#2_dim }
483                             \l__enumext_current_widest_dim
484                         },
485         label .initial:n = #3,
486         label .value_required:n = true,
487         ref   .code:n    = \__enumext_set_label_ref_h:n {##1},
488         ref   .value_required:n = true,
489       }
490   }
491 \__enumext_tmp:nnn { enumext* } {  vii } { \arabic*.}
492 \__enumext_tmp:nnn { keyans*  } { viii } { (\Alph*) }
```

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

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

label

\l__enumext_label_v_tl

\l__enumext_label_vi_tl

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

```
493 \keys_define:nn { enumext / keyans }
494   {
495     label .code:n    = {
496                       \__enumext_label_style:cvn { l__enumext_label_v_tl }
497                         { l__enumext_counter_v_tl } {#1}
498                       \dim_set_eq:cN  { l__enumext_labelwidth_v_dim }
499                         \l__enumext_current_widest_dim
500                       \__enumext_label_style:cvn { l__enumext_label_vi_tl }
501                         { l__enumext_counter_vi_tl } {#1}
502                       \dim_set_eq:cN  { l__enumext_labelwidth_v_dim }
503                         \l__enumext_current_widest_dim
504                     },
505     label .initial:n = (\Alph*),
506     label .value_required:n = true,
507   }
```

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

10.12 Setting start and widest keys

__enumext_start_from:NNn

__enumext_start_from:ccn

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

#1: `\l__enumext_label_X_tl`

#2: `\l__enumext_start_X_int`

#3: ⟨*integer or string*⟩

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

```
508 \cs_new_protected:Npn \__enumext_start_from:NNn #1 #2 #3
509   {
510     \__enumext_if_is_int:nTF { #3 }
511       {
512         \int_set:Nn #2 {#3}
513       }
514       {
515         \regex_match:nVT { \c{Alph} | \c{alph} } {#1}
516           { \int_set:Nn #2 { \int_from_alph:n {#3} } }
517         \regex_match:nVT { \c{Roman} | \c{roman} } {#1}
518           { \int_set:Nn #2  { \int_from_roman:n {#3} } }
519       }
520   }
521 \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.

```
522  \cs_new_protected:Npn \__enumext_widest_from:nNNn #1 #2 #3 #4
523    {
524      \__enumext_if_is_int:nTF {#4}
525        {
526          \setcounter{enumX#1} { #4 }
527        }
528        {
529          \regex_match:nVT { \c{Alph} | \c{alph} } {#2}
530            { \setcounter{enumX#1} { \int_from_alph:n {#4} } }
531          \regex_match:nVT { \c{Roman} | \c{roman} } {#2}
532            { \setcounter{enumX#1} { \int_from_roman:n {#4} } }
533        }
534      \__enumext_label_width_by_box:cv
535        { l__enumext_labelwidth_#1_dim } { l__enumext_label_#1_tl }
536    }
537  \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.

```
538  \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
539    {
540      \keys_define:nn { enumext / #1 }
541        {
542          start  .code:n     = {
543                                \__enumext_start_from:ccn
544                                  { l__enumext_label_#2_tl }
545                                  { l__enumext_start_#2_int } {##1}
546                              },
547          start  .initial:n = 1,
548          widest .code:n     = {
549                                \__enumext_widest_from:nccn {#2}
550                                  { l__enumext_label_#2_tl }
551                                  { l__enumext_labelwidth_#2_dim } {##1}
552                              },
553          widest .value_required:n = true,
554          start  .value_required:n = true,
555        }
556    }
557  \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }
```

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

10.13 Setting keys for vertical spaces

topsep
partopsep
parsep
noitemsep
nosep

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

```
558  \cs_set_protected:Npn \__enumext_tmp:nnnnnn #1 #2 #3 #4 #5 #6
559    {
560      \keys_define:nn { enumext / #1 }
561        {
562          topsep    .skip_set:c = { l__enumext_topsep_#2_skip },
563          topsep    .initial:n  = {#3},
564          topsep    .value_required:n = true,
565          partopsep .skip_set:c = { l__enumext_partopsep_#2_skip },
566          partopsep .initial:n  = {#4},
567          partopsep .value_required:n = true,
568          parsep    .skip_set:c = { l__enumext_parsep_#2_skip },
569          parsep    .initial:n  = {#5},
```

```
570        parsep    .value_required:n = true,
571        itemsep   .skip_set:c = { l__enumext_itemsep_#2_skip },
572        itemsep   .initial:n  = {#6},
573        itemsep   .value_required:n = true,
574        noitemsep .meta:n     = { itemsep = 0pt, parsep = 0pt },
575        noitemsep .value_forbidden:n = true,
576        nosep     .meta:n     = {
577                               itemsep = 0pt, parsep= 0pt,
578                               topsep = 0pt, partopsep = 0pt,
579                             },
580        nosep     .value_forbidden:n = true,
581      }
582    }
```

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

```
583 \__enumext_tmp:nnnnnn { level-1 } { i } { 8.0pt plus 2.0pt minus 4.0pt }
584    { 2.0pt plus 1.0pt minus 1.0pt } { 4.0pt plus 2.0pt minus 1.0pt }
585    { 4.0pt plus 2.0pt minus 1.0pt }
586 \__enumext_tmp:nnnnnn { level-2 } { ii } { 4.0pt plus 2.0pt minus 1.0pt }
587    { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
588    { 2.0pt plus 1.0pt minus 1.0pt }
589 \__enumext_tmp:nnnnnn { level-3 } { iii } { 2.0pt plus 1.0pt minus 1.0pt }
590    { 1.0pt minus 1.0pt }{ 0pt }{ 2.0pt plus 1.0pt minus 1.0pt }
591 \__enumext_tmp:nnnnnn { level-4 } { iv } { 2.0pt plus 1.0pt minus 1.0pt }
592    { 1.0pt minus 1.0pt }{ 0pt }{ 2.0pt plus 1.0pt minus 1.0pt }
593 \__enumext_tmp:nnnnnn { keyans  } { v }{ 4.0pt plus 2.0pt minus 1.0pt }
594    { 2.0pt plus 1.0pt minus 1.0pt }{ 2.0pt plus 1.0pt minus 1.0pt }
595    { 2.0pt plus 1.0pt minus 1.0pt }
596 \__enumext_tmp:nnnnnn { enumext* } { vii } { 8.0pt plus 2.0pt minus 4.0pt }
597    { 2.0pt plus 1.0pt minus 1.0pt } { 4.0pt plus 2.0pt minus 1.0pt }
598    { 4.0pt plus 2.0pt minus 1.0pt }
599 \__enumext_tmp:nnnnnn { keyans* } { viii } { 4.0pt plus 2.0pt minus 1.0pt }
600    { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
601    { 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.

```
602 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
603   {
604     \keys_define:nn { enumext / #1 }
605       {
606         itemindent   .dim_set:c = { l__enumext_fake_item_indent_#2_dim },
607         itemindent   .value_required:n = true,
608         rightmargin   .dim_set:c = { l__enumext_rightmargin_#2_dim },
609         rightmargin   .value_required:n = true,
610         listparindent .dim_set:c = { l__enumext_listparindent_#2_dim },
611         listparindent .value_required:n = true,
612         list-offset   .dim_set:c = { l__enumext_listoffset_#2_dim },
613         list-offset   .value_required:n = true,
614         list-indent   .code:n    =
615                           \bool_set_true:c { l__enumext_leftmargin_tmp_#2_bool }
616                           \dim_set:cn { l__enumext_leftmargin_tmp_#2_dim } {##1},
617         list-indent   .value_required:n = true,
618       }
619   }
620 \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.

```
621 \cs_set_protected:Npn \__enumext_tmp:n #1
622   {
623     \keys_define:nn { enumext / #1 } { list-indent .initial:n  = 0pt, }
624   }
625 \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.

```
626  \cs_set_protected:Nn \__enumext_fake_item:
627    {
628      \dim_compare:nNnT
629        { \dim_use:c { l__enumext_fake_item_indent_ \__enumext_level: _dim } }
630        >
631        { \c_zero_dim }
632        {
633          \tl_set:ce { l__enumext_fake_item_indent_ \__enumext_level: _tl }
634            {
635              \exp_not:N \mode_leave_vertical:
636              \exp_not:n { \skip_horizontal:n }
637                { \dim_use:c { l__enumext_fake_item_indent_ \__enumext_level: _dim } }
638              \ignorespaces
639            }
640        }
641    }
642  \cs_set_protected:Nn \__enumext_keyans_fake_item:
643    {
644      \dim_compare:nNnT
645        { \l__enumext_fake_item_indent_v_dim } > { \c_zero_dim }
646        {
647          \tl_set:Ne \l__enumext_fake_item_indent_v_tl
648            {
649              \exp_not:N \mode_leave_vertical:
650              \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_v_dim
651            }
652        }
653    }
654  \cs_set_protected:Nn \__enumext_fake_item_vii:
655    {
656      \dim_compare:nNnT
657        { \l__enumext_fake_item_indent_vii_dim } > { \c_zero_dim }
658        {
659          \tl_set:Ne \l__enumext_fake_item_indent_vii_tl
660            {
661              \exp_not:N \mode_leave_vertical:
662              \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_vii_dim
663            }
664        }
665    }
666  \cs_set_protected:Nn \__enumext_fake_item_viii:
667    {
668      \dim_compare:nNnT
669        { \l__enumext_fake_item_indent_viii_dim } > { \c_zero_dim }
670        {
671          \tl_set:Ne \l__enumext_fake_item_indent_viii_tl
672            {
673              \exp_not:N \mode_leave_vertical:
674              \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_viii_dim
675            }
676        }
677    }
```

(*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.

```
678  \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
679    {
680      \keys_define:nn { enumext / #1 }
681        {
682          show-length .bool_set:c = { l__enumext_show_length_#2_bool },
```

```
683          show-length .initial:n  = false,
684        }
685    }
686  \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.

```
687  \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
688    {
689      \keys_define:nn { enumext / #1 }
690        {
691          before  .tl_set:c   = { l__enumext_before_no_starred_key_#2_tl },
692          before  .value_required:n = true,
693          before* .tl_set:c   = { l__enumext_before_starred_key_#2_tl },
694          before* .value_required:n = true,
695          after   .tl_set:c   = { l__enumext_after_stop_list_#2_tl },
696          after   .value_required:n = true,
697          first   .tl_set:c   = { l__enumext_after_list_args_#2_tl },
698          first   .value_required:n = true,
699        }
700    }
701  \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.

```
702  \cs_new_protected:Nn \__enumext_before_args_exec:
703    {
704      \tl_use:c { l__enumext_before_starred_key_ \__enumext_level: _tl }
705    }
```

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*⟩.

```
706  \cs_new_protected:Nn \__enumext_before_keys_exec:
707    {
708      \tl_use:c { l__enumext_before_no_starred_key_ \__enumext_level: _tl }
709    }
```

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

```
710  \cs_new_protected:Nn \__enumext_after_stop_list:
711    {
712      \tl_use:c { l__enumext_after_stop_list_ \__enumext_level: _tl }
713    }
```

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.

```
714  \cs_new_protected:Nn \__enumext_after_args_exec:
715    {
716      \tl_use:c { l__enumext_after_list_args_ \__enumext_level: _tl }
717    }
```

(*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.

```
718  \cs_new_protected:Nn \__enumext_before_args_exec_v:
719    {
720      \tl_use:N \l__enumext_before_starred_key_v_tl
721    }
```

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*⟩.

```
722 \cs_new_protected:Nn \__enumext_before_keys_exec_v:
723   {
724     \tl_use:N \l__enumext_before_no_starred_key_v_tl
725   }
```

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

```
726 \cs_new_protected:Nn \__enumext_after_stop_list_v:
727   {
728     \tl_use:N \l__enumext_after_stop_list_v_tl
729   }
```

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.

```
730 \cs_new_protected:Nn \__enumext_after_args_exec_v:
731   {
732     \tl_use:N \l__enumext_after_list_args_v_tl
733   }
```

(*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*

<div style="float:left">

__enumext_before_args_exec_vii:
__enumext_before_keys_exec_vii
__enumext_after_stop_list_vii:
__enumext_after_args_exec_vii:

</div>

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.

```
734 \cs_new_protected:Nn \__enumext_before_args_exec_vii:
735   {
736     \tl_use:N \l__enumext_before_starred_key_vii_tl
737   }
738 \cs_new_protected:Nn \__enumext_before_args_exec_viii:
739   {
740     \tl_use:N \l__enumext_before_starred_key_viii_tl
741   }
```

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*⟩.

```
742 \cs_new_protected:Nn \__enumext_before_keys_exec_vii:
743   {
744     \tl_use:N \l__enumext_before_no_starred_key_vii_tl
745   }
746 \cs_new_protected:Nn \__enumext_before_keys_exec_viii:
747   {
748     \tl_use:N \l__enumext_before_no_starred_key_viii_tl
749   }
```

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

```
750 \cs_new_protected:Nn \__enumext_after_stop_list_vii:
751   {
752     \tl_use:N \l__enumext_after_stop_list_vii_tl
753   }
754 \cs_new_protected:Nn \__enumext_after_stop_list_viii:
755   {
756     \tl_use:N \l__enumext_after_stop_list_viii_tl
757   }
```

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.

```
758 \cs_new_protected:Nn \__enumext_after_args_exec_vii:
759   {
760     \tl_use:N \l__enumext_after_list_args_vii_tl
761   }
762 \cs_new_protected:Nn \__enumext_after_args_exec_viii:
763   {
764     \tl_use:N \l__enumext_after_list_args_viii_tl
765   }
```

(*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.

```
766 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
767   {
768     \keys_define:nn { enumext / #1 }
769       {
770         mini-env    .dim_set:c  = { l__enumext_minipage_right_#2_dim },
771         mini-env    .value_required:n = true,
772         mini-sep    .dim_set:c  = { l__enumext_minipage_hsep_#2_dim },
773         mini-sep    .initial:n  = 0.3333em,
774         mini-sep    .value_required:n = true,
775         columns-sep .dim_set:c  = { l__enumext_columns_sep_#2_dim },
776         columns-sep .value_required:n = true,
777         columns     .int_set:c  = { l__enumext_columns_#2_int },
778         columns     .initial:n  = 1,
779         columns     .value_required:n = true,
780       }
781   }
782 \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`.

```
783 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
784   {
785     \keys_define:nn { enumext / #1 }
786       {
787         columns    .initial:n = 2,
788         miniright  .tl_gset:c = { g__enumext_miniright_code_#2_tl },
789         miniright  .value_required:n = true,
790         miniright* .code:n    = {
791                                   \bool_gset_true:c { g__enumext_minipage_center_#2_bool }
792                                   \keys_set:nn { enumext / #1 } { miniright = {##1} }
793                                 },
794         miniright* .value_required:n = true,
795       }
796   }
797 \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.
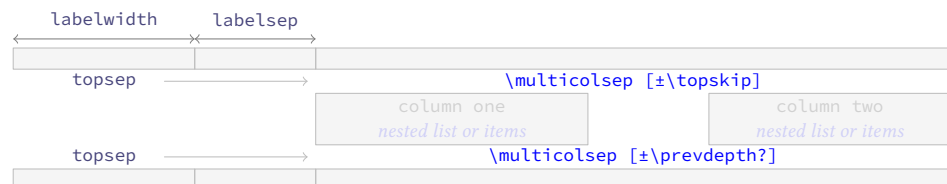
Figure 7: Representation of the vertical space in `multicols` for a nested level.

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*.

```
798  \cs_new_protected:Nn \__enumext_multi_set_vskip:
799    {
800      \skip_set:cn { l__enumext_multicols_above_ \__enumext_level: _skip }
801        {
802          \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
803        }
804      \skip_set:cn { l__enumext_multicols_below_ \__enumext_level: _skip }
805        {
806          \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
807        }
808      \__enumext_add_pre_parsep:
809    }
```

(*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*.

```
810  \cs_new_protected:Nn \__enumext_add_pre_parsep:
811    {
812      \int_case:nn { \l__enumext_level_int }
813        {
814          { 2 }{
815              \skip_if_eq:nnF { \l__enumext_parsep_i_skip } { \c_zero_skip }
816                {
817                  \skip_add:Nn \l__enumext_multicols_above_ii_skip { \l__enumext_parsep_i_skip }
818                }
819            }
820          { 3 }{
821              \skip_if_eq:nnF { \l__enumext_parsep_ii_skip } { \c_zero_skip }
822                {
823                  \skip_add:Nn \l__enumext_multicols_above_iii_skip { \l__enumext_parsep_ii_skip
824                }
825            }
826          { 4 }{
827              \skip_if_eq:nnF { \l__enumext_parsep_iii_skip } { \c_zero_skip }
828                {
829                  \skip_add:Nn \l__enumext_multicols_above_iv_skip { \l__enumext_parsep_iii_skip
830                }
831            }
832        }
833    }
```

(*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*⟩.

```
834  \cs_new_protected:Nn \__enumext_multi_addvspace:
835    {
836      \__enumext_multi_set_vskip:
837      \mode_if_vertical:T
838        {
839          \skip_add:cn { l__enumext_multicols_above_ \__enumext_level: _skip }
840            {
841              \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
842            }
843          \skip_add:cn { l__enumext_multicols_below_ \__enumext_level: _skip }
844            {
845              \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
846            }
847        }
```

```
848        \par\nopagebreak
849        \addvspace{ \skip_use:c { l__enumext_multicols_above_ \__enumext_level: _skip } }
850      }
```

(*End of definition for* __enumext_multi_addvspace:.)

10.18.2 Adjustment of vertical spaces for multicols in keyans

__enumext_keyans_multi_set_vskip:
__enumext_keyans_multi_addvspace:

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

```
851  \cs_new_protected:Nn \__enumext_keyans_multi_set_vskip:
852    {
853      \skip_set:Nn \l__enumext_multicols_above_v_skip
854        {
855          \l__enumext_topsep_v_skip
856        }
857      \skip_set:Nn \l__enumext_multicols_below_v_skip
858        {
859          \l__enumext_topsep_v_skip
860        }
861    }
862  \cs_new_protected:Nn \__enumext_keyans_multi_addvspace:
863    {
864      \__enumext_keyans_multi_set_vskip:
865      \mode_if_vertical:T
866        {
867          \skip_add:Nn \l__enumext_multicols_above_v_skip
868            {
869              \skip_use:N \l__enumext_partopsep_v_skip
870            }
871          \skip_add:Nn \l__enumext_multicols_below_v_skip
872            {
873              \skip_use:N \l__enumext_partopsep_v_skip
874            }
875        }
876      \par\nopagebreak
877      \addvspace{ \l__enumext_multicols_above_v_skip }
878    }
```

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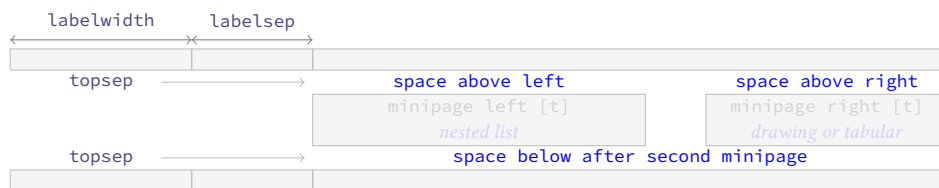
Figure 8: Representation of the minipage spacing adjustment for a nested level.

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.

```
879  \DeclareDocumentEnvironment{__enumext_mini_env*}{ m }
880    {
881      \__enumext_minipage:w [ t ] { #1 }
882        \legacy_if_gset_false:n { @minipage }
883        \vspace { 0pt }
884    }
885    { \__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`.

```
886  \cs_new_protected:Nn \__enumext_mini_set_vskip:
887    {
888      \int_compare:nNnTF
889        { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } > { 1 }
890        {
```

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.

```
891              \skip_if_eq:nnTF
892                { \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip } } { \c_zero_skip }
893                {
894                  \skip_set:Nn \l__enumext_minipage_left_skip
895                    {
896                      -0.150\box_dp:N \strutbox
897                    }
898                  \skip_set:Nn \l__enumext_minipage_right_skip
899                    {
900                      0.695\box_dp:N \strutbox
901                    }
902                  \skip_set:Nn \l__enumext_minipage_after_skip
903                    {
904                      \box_dp:N \strutbox
905                    }
906                  \__enumext_zero_parsep:
907                }
908                {
909                  \skip_set:Nn \l__enumext_minipage_left_skip
910                    {
911                      \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
912                    }
913                  \skip_set:Nn \l__enumext_minipage_right_skip
914                    {
915                      0.695\box_dp:N \strutbox
916                    }
917                  \skip_set:Nn \l__enumext_minipage_after_skip
918                    {
919                      1.85\box_dp:N \strutbox
920                      + \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
921                    }
922                }
923              }
924            {
```

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.

```
925              \skip_if_eq:nnTF
926                { \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip } } { \c_zero_skip }
927                {
928                  \skip_set:Nn \l__enumext_minipage_left_skip
```

```
929                        {
930                            0.5\box_dp:N \strutbox
931                            - \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
932                        }
933                    \skip_set:Nn \l__enumext_minipage_right_skip
934                        {
935                            \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
936                        }
937                    \skip_set:Nn \l__enumext_minipage_after_skip
938                        {
939                            1.6\box_dp:N \strutbox
940                        }
941                }
942                {
943                    \skip_set:Nn \l__enumext_minipage_left_skip
944                        {
945                            0.5875\box_dp:N \strutbox
946                            - \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
947                        }
948                    \skip_set:Nn \l__enumext_minipage_right_skip
949                        {
950                            + \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
951                            + \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
952                        }
953                    \skip_set:Nn \l__enumext_minipage_after_skip
954                        {
955                            0.325\box_dp:N \strutbox
956                            + \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
957                        }
958                }
959            }
960        }
```

(*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.

```
961 \cs_new_protected:Nn \__enumext_zero_parsep:
962    {
963        \int_case:nn { \l__enumext_level_int }
964            {
965            { 2 }{
966                    \skip_if_eq:nnF { \l__enumext_parsep_i_skip } { \c_zero_skip }
967                        {
968                            \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
969                        }
970                }
971            { 3 }{
972                    \skip_if_eq:nnF { \l__enumext_parsep_ii_skip } { \c_zero_skip }
973                        {
974                            \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
975                        }
976                }
977            { 4 }{
978                    \skip_if_eq:nnF { \l__enumext_parsep_iii_skip } { \c_zero_skip }
979                        {
980                            \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
981                        }
982                }
983            }
984    }
```

(*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*.

```
985 \cs_new_protected:Nn \__enumext_mini_addvspace:
986    {
```

```
987     \__enumext_mini_set_vskip:
988     \mode_if_vertical:T
989       {
990         \skip_add:Nn \l__enumext_minipage_left_skip
991           {
992             \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
993           }
994         \skip_add:Nn \l__enumext_minipage_after_skip
995           {
996             \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
997           }
998       }
999     \par\nopagebreak
1000    \addvspace { \l__enumext_minipage_left_skip }
1001   }
```

(*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`.

```
1002  \cs_new_protected:Nn \__enumext_keyans_mini_set_vskip:
1003    {
1004      \skip_zero_new:N \l__enumext_minipage_after_skip
1005      \skip_zero_new:N \l__enumext_minipage_left_skip
1006      \skip_zero_new:N \l__enumext_minipage_right_skip
1007      \int_compare:nNnTF { \l__enumext_columns_v_int } > { 1 }
1008        {
1009          \skip_if_eq:nnTF { \l__enumext_topsep_v_skip } { \c_zero_skip }
1010            {
1011              \skip_set:Nn \l__enumext_minipage_left_skip { -0.25\box_dp:N \strutbox }
1012              \skip_set:Nn \l__enumext_minipage_right_skip { 0.705\box_dp:N \strutbox }
1013              \skip_set:Nn \l__enumext_minipage_after_skip { \box_dp:N \strutbox }
1014              \skip_if_eq:nnF { \l__enumext_parsep_i_skip } { \c_zero_skip }
1015                {
1016                  \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
1017                }
1018            }
1019            {
1020              \skip_set:Nn \l__enumext_minipage_left_skip
1021                {
1022                  \skip_use:N \l__enumext_topsep_v_skip
1023                }
1024              \skip_set:Nn \l__enumext_minipage_right_skip
1025                {
1026                  0.705\box_dp:N \strutbox
1027                }
1028              \skip_set:Nn \l__enumext_minipage_after_skip
1029                {
1030                  1.85\box_dp:N \strutbox + \l__enumext_topsep_v_skip
1031                }
1032            }
1033        }
1034        {
1035          \skip_if_eq:nnTF { \l__enumext_topsep_v_skip } { \c_zero_skip }
1036            {
1037              \skip_set:Nn \l__enumext_minipage_left_skip
1038                {
1039                  0.5\box_dp:N \strutbox
1040                  + \l__enumext_partopsep_v_skip
1041                }
1042              \skip_set:Nn \l__enumext_minipage_right_skip
1043                {
1044                  \l__enumext_partopsep_v_skip
1045                }
1046              \skip_set:Nn \l__enumext_minipage_after_skip { 1.6\box_dp:N \strutbox }
1047            }
1048            {
1049              \skip_set:Nn \l__enumext_minipage_left_skip
1050                {
```

```
1051                      0.5875\box_dp:N \strutbox - \l__enumext_partopsep_v_skip
1052                  }
1053              \skip_set:Nn \l__enumext_minipage_right_skip
1054                  {
1055                      \l__enumext_topsep_v_skip + \l__enumext_partopsep_v_skip
1056                  }
1057              \skip_set:Nn \l__enumext_minipage_after_skip
1058                  {
1059                      0.325\box_dp:N \strutbox + \l__enumext_topsep_v_skip
1060                  }
1061          }
1062      }
1063  }
```

(*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.

```
1064  \cs_new_protected:Nn \__enumext_keyans_mini_addvspace:
1065      {
1066          \__enumext_keyans_mini_set_vskip:
1067          \mode_if_vertical:T
1068              {
1069                  \skip_add:Nn \l__enumext_minipage_left_skip
1070                      {
1071                          \l__enumext_partopsep_v_skip
1072                      }
1073                  \skip_add:Nn \l__enumext_minipage_after_skip
1074                      {
1075                          \l__enumext_partopsep_v_skip
1076                      }
1077              }
1078          \par\nopagebreak
1079          \addvspace { \l__enumext_minipage_left_skip }
1080      }
```

(*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*.

```
1081  \cs_new_protected:Nn \__enumext_mini_set_vskip_vii:
1082      {
1083          \skip_zero_new:N \l__enumext_minipage_left_skip
1084          \skip_gzero_new:N \g__enumext_minipage_right_skip
1085          \skip_gzero_new:N \g__enumext_minipage_after_skip
1086          \skip_if_eq:nnTF { \l__enumext_topsep_vii_skip } { \c_zero_skip }
1087              {
1088                  \skip_set:Nn \l__enumext_minipage_left_skip { 0.5\box_dp:N \strutbox }
1089                  \skip_gset:Nn \g__enumext_minipage_right_skip { 0.325\box_dp:N \strutbox }
1090              }
1091              {
1092                  \skip_set:Nn \l__enumext_minipage_left_skip { 0.5875\box_dp:N \strutbox }
1093                  \skip_gset:Nn \g__enumext_minipage_right_skip
1094                      {
1095                          \l__enumext_topsep_vii_skip
1096                      }
1097                  \skip_gset:Nn \g__enumext_minipage_after_skip
1098                      {
1099                          0.325\box_dp:N \strutbox + \l__enumext_topsep_vii_skip
1100                      }
1101              }
1102      }
1103  \cs_new_protected:Nn \__enumext_mini_set_vskip_viii:
1104      {
1105          \skip_zero_new:N \l__enumext_minipage_after_skip
```

```
1106    \skip_zero_new:N \l__enumext_minipage_left_skip
1107    \skip_zero_new:N \l__enumext_minipage_right_skip
1108    \skip_if_eq:nnTF { \l__enumext_topsep_viii_skip } { \c_zero_skip }
1109      {
1110        \skip_set:Nn \l__enumext_minipage_left_skip
1111          {
1112            0.5\box_dp:N \strutbox
1113          }
1114        \skip_set:Nn \l__enumext_minipage_right_skip
1115          {
1116            \l__enumext_partopsep_viii_skip
1117          }
1118        \skip_set:Nn \l__enumext_minipage_after_skip
1119          {
1120            1.6\box_dp:N \strutbox
1121          }
1122      }
1123      {
1124        \skip_set:Nn \l__enumext_minipage_left_skip
1125          {
1126            0.5875\box_dp:N \strutbox
1127          }
1128        \skip_set:Nn \l__enumext_minipage_right_skip
1129          {
1130            \l__enumext_topsep_viii_skip
1131          }
1132        \skip_set:Nn \l__enumext_minipage_after_skip
1133          {
1134            0.325\box_dp:N \strutbox + \l__enumext_topsep_viii_skip
1135          }
1136      }
1137    }
```

(*End of definition for* __enumext_mini_set_vskip_vii: *and* __enumext_mini_set_vskip_viii:.)

__enumext_mini_addvspace_vii: The functions __enumext_mini_addvspace_vii: and __enumext_mini_addvspace_viii: will
__enumext_mini_addvspace_viii: 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.

```
1138  \cs_new_protected:Nn \__enumext_mini_addvspace_vii:
1139    {
1140      \__enumext_mini_set_vskip_vii:
1141      \par\nopagebreak
1142      \addvspace { \l__enumext_minipage_left_skip }
1143    }
1144  \cs_new_protected:Nn \__enumext_mini_addvspace_viii:
1145    {
1146      \__enumext_mini_set_vskip_viii:
1147      \par\nopagebreak
1148      \addvspace { \l__enumext_minipage_left_skip }
1149    }
```

(*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.

```
1150  \NewDocumentCommand \miniright { s }
1151    {
1152      \int_compare:nNnT { \l__enumext_keyans_pic_level_int } = { 1 }
1153        {
1154          \msg_error:nnn { enumext } { wrong-miniright-place }
```

```
1155              }
1156          \int_compare:nNnT { \l__enumext_level_int } = { 0 }
1157            {
1158              \msg_error:nnn { enumext } { wrong-miniright-place }
1159            }
1160          \int_compare:nNnTF { \l__enumext_keyans_level_int } = { 1 }
1161            {
1162              \__enumext_keyans_mini_right_cmd:n {#1}
1163            }
1164          { \__enumext_mini_right_cmd:n {#1} }
1165        }
```

(*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.

```
1166  \cs_new_protected:Npn \__enumext_mini_right_cmd:n #1
1167    {
1168      \dim_compare:nNnTF
1169        { \dim_use:c { l__enumext_minipage_right_ \__enumext_level: _dim } } > { \c_zero_dim }
1170        {
1171          \__enumext_multicols_stop:
1172          \end{__enumext_mini_env*}
1173          \hfill
1174          \begin{__enumext_mini_env*}
1175            { \dim_use:c { l__enumext_minipage_right_ \__enumext_level: _dim } }
1176            \par\addvspace { \l__enumext_minipage_right_skip }
1177            \bool_if:nF {#1}
1178              {
1179                \centering
1180              }
1181            \int_gzero:N \g__enumext_minipage_stat_int
1182        }
1183        { \msg_error:nnn { enumext } { wrong-miniright-use } }
1184    }
```

(*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.

```
1185  \cs_new_protected:Npn \__enumext_keyans_mini_right_cmd:n #1
1186    {
1187      \dim_compare:nNnTF { \l__enumext_minipage_right_v_dim } > { \c_zero_dim }
1188        {
1189          \__enumext_keyans_multicols_stop:
1190          \end{__enumext_mini_env*}
1191          \hfill
1192          \begin{__enumext_mini_env*}{ \l__enumext_minipage_right_v_dim }
1193            \par\addvspace { \l__enumext_minipage_right_skip }
1194            \bool_if:nF {#1}
1195              {
1196                \centering
1197              }
1198            \int_gzero:N \g__enumext_minipage_stat_int
1199        }
1200        { \msg_error:nnn { enumext } { wrong-miniright-use } }
1201    }
```

(*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.

```
1202 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1203   {
1204     \keys_define:nn { enumext / #1 }
1205       {
1206         above  .skip_set:c = { l__enumext_vspace_above_#2_skip },
1207         above  .value_required:n = true,
1208         above* .code:n     = \bool_set_true:c { l__enumext_vspace_a_star_#2_bool }
1209                              \keys_set:nn { enumext / #1 } { above = {##1} },
1210         above* .value_required:n = true,
1211         below  .skip_set:c = { l__enumext_vspace_below_#2_skip },
1212         below  .value_required:n = true,
1213         below* .code:n     = \bool_set_true:c { l__enumext_vspace_b_star_#2_bool }
1214                              \keys_set:nn { enumext / #1 } { below = {##1} },
1215         below* .value_required:n = true,
1216       }
1217   }
1218 \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.

```
1219 \cs_new_protected:Nn \__enumext_vspace_above:
1220   {
1221     \skip_if_eq:nnF
1222       { \skip_use:c { l__enumext_vspace_above_ \__enumext_level: _skip } } { \c_zero_skip }
1223       {
1224         \bool_if:cTF { l__enumext_vspace_a_star_ \__enumext_level: _bool }
1225           {
1226             \vspace*{ \skip_use:c { l__enumext_vspace_above_ \__enumext_level: _skip } }
1227           }
1228           {
1229             \vspace { \skip_use:c { l__enumext_vspace_above_ \__enumext_level: _skip } }
1230           }
1231       }
1232   }
```

(*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.

```
1233 \cs_new_protected:Nn \__enumext_vspace_below:
1234   {
1235     \skip_if_eq:nnF
1236       { \skip_use:c { l__enumext_vspace_below_ \__enumext_level: _skip } } { \c_zero_skip }
1237       {
1238         \bool_if:cTF { l__enumext_vspace_b_star_ \__enumext_level: _bool }
1239           {
1240             \vspace*{ \skip_use:c { l__enumext_vspace_below_ \__enumext_level: _skip } }
1241           }
1242           {
1243             \vspace { \skip_use:c { l__enumext_vspace_below_ \__enumext_level: _skip } }
1244           }
1245       }
1246   }
```

(*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.

```
1247 \cs_new_protected:Nn \__enumext_vspace_above_v:
1248   {
1249     \skip_if_eq:nnF { \l__enumext_vspace_above_v_skip } { \c_zero_skip }
1250       {
1251         \bool_if:NTF \l__enumext_vspace_a_star_v_bool
1252           {
1253             \vspace*{ \l__enumext_vspace_above_v_skip }
1254           }
1255           { \vspace { \l__enumext_vspace_above_v_skip } }
1256       }
1257   }
```

(*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.

```
1258 \cs_new_protected:Nn \__enumext_vspace_below_v:
1259   {
1260     \skip_if_eq:nnF { \l__enumext_vspace_below_v_skip } { \c_zero_skip }
1261       {
1262         \bool_if:NTF \l__enumext_vspace_b_star_v_bool
1263           {
1264             \vspace*{ \l__enumext_vspace_below_v_skip }
1265           }
1266           { \vspace { \l__enumext_vspace_below_v_skip } }
1267       }
1268   }
```

(*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.

```
1269 \cs_new_protected:Nn \__enumext_vspace_above_vii:
1270   {
1271     \skip_if_eq:nnF { \l__enumext_vspace_above_vii_skip } { \c_zero_skip }
1272       {
1273         \bool_if:NTF \l__enumext_vspace_a_star_vii_bool
1274           {
1275             \vspace*{ \l__enumext_vspace_above_vii_skip }
1276           }
1277           { \vspace { \l__enumext_vspace_above_vii_skip } }
1278       }
1279   }
1280 \cs_new_protected:Nn \__enumext_vspace_above_viii:
1281   {
1282     \skip_if_eq:nnF { \l__enumext_vspace_above_viii_skip } { \c_zero_skip }
1283       {
1284         \bool_if:NTF \l__enumext_vspace_a_star_viii_bool
1285           {
1286             \vspace*{ \l__enumext_vspace_above_viii_skip }
1287           }
1288           { \vspace { \l__enumext_vspace_above_viii_skip } }
1289       }
1290   }
```

(*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.

```
1291 \cs_new_protected:Nn \__enumext_vspace_below_vii:
1292   {
1293     \skip_if_eq:nnF { \l__enumext_vspace_below_vii_skip } { \c_zero_skip }
1294       {
1295         \bool_if:NTF \l__enumext_vspace_b_star_vii_bool
1296           {
1297             \vspace*{ \l__enumext_vspace_below_vii_skip }
```

```
1298                    }
1299                    { \vspace { \l__enumext_vspace_below_vii_skip } }
1300                }
1301            }
1302   \cs_new_protected:Nn \__enumext_vspace_below_viii:
1303        {
1304            \skip_if_eq:nnF { \l__enumext_vspace_below_viii_skip } { \c_zero_skip }
1305                {
1306                    \bool_if:NTF \l__enumext_vspace_b_star_viii_bool
1307                        {
1308                            \vspace*{ \l__enumext_vspace_below_viii_skip }
1309                        }
1310                        { \vspace { \l__enumext_vspace_below_viii_skip } }
1311                }
1312        }
```

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

10.21 Setting save-ans and resume keys

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

save-ans
resume
resume*

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

```
1313   \keys_define:nn { enumext / level-1 }
1314        {
1315            save-ans .code:n = \__enumext_storing_set:n {#1},
1316            save-ans .value_required:n = true,
1317            resume   .code:n = \__enumext_resume_counter:,
1318            resume   .value_forbidden:n = true,
1319            resume*  .code:n = \__enumext_resume_counter_star:,
1320            resume*  .value_forbidden:n = true,
1321        }

1322   \keys_define:nn { enumext / enumext* }
1323        {
1324            save-ans .code:n = \__enumext_storing_set:n {#1},
1325            save-ans .value_required:n = true,
1326            resume   .code:n = \__enumext_resume_counter_vii:,
1327            resume   .value_forbidden:n = true,
1328        }
```

(*End of definition for* save-ans *,* resume *, and* resume*.)

__enumext_storing_set:n

The function __enumext_storing_set:n executed by the save-ans key sets the parameters for the operation of \anskey, keyans and keyanspic. The variable \l__enumext_store_name_tl will have the *"store name"* with which the ⟨*sequence*⟩ and ⟨*prop list*⟩ will be created, if it does not exist it will create it globally.

The boolean var \l__enumext_store_active_bool will be set to true activating the entire internal *storage mechanism*, then the integer variable for the resume key will be created (if not exist), finally the function __enumext_check_ans_int:n will be called to activate the internal mechanism for checking the answers if the boolean variable \l__enumext_check_ans_bool set by check-ans key are active.

```
1329   \cs_new_protected:Npn \__enumext_storing_set:n #1
1330        {
1331            \tl_set:Ne \l__enumext_store_name_tl {#1}
1332            \prop_if_exist:cF { g__enumext_ \l__enumext_store_name_tl _prop }
1333                {
1334                    \prop_new:c { g__enumext_ \l__enumext_store_name_tl _prop }
1335                }
1336            \seq_if_exist:cF { g__enumext_ \l__enumext_store_name_tl _seq }
1337                {
1338                    \seq_new:c { g__enumext_ \l__enumext_store_name_tl _seq }
1339                }
1340            \bool_set_true:N \l__enumext_store_active_bool
1341            \bool_set_true:N \l__enumext_store_ans_bool
1342            \int_if_exist:cF { g__enumext_resume_#1_int }
1343                {
1344                    \int_new:c { g__enumext_resume_#1_int }
1345                }
1346        }
```

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

__enumext_resume_counter:
__enumext_resume_counter_vii:

The functions `\__enumext_resume_counter:` and `\__enumext_resume_counter_vii:` used by `resume` key in `enumext` and `enumext*`. If `save-ans` key present then set the start value from integer created by `\__enumext_storing_set:n`.

```
1347 \cs_new_protected:Nn \__enumext_resume_counter:
1348   {
1349     \bool_if:NT \l__enumext_store_active_bool
1350       {
1351         \int_gset:Nn \g__enumext_resume_int
1352           {
1353             \int_use:c { g__enumext_resume_ \l__enumext_store_name_tl _int }
1354           }
1355       }
1356     \bool_set_true:N \l__enumext_resume_bool
1357   }
1358 \cs_new_protected:Nn \__enumext_resume_counter_vii:
1359   {
1360     \bool_if:NT \l__enumext_store_active_bool
1361       {
1362         \int_gset:Nn \g__enumext_resume_int
1363           {
1364             \int_use:c { g__enumext_resume_ \l__enumext_store_name_tl _int }
1365           }
1366       }
1367     \bool_set_true:N \l__enumext_resume_vii_bool
1368   }
```

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

10.22 Setting check-ans key

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 increment 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_all_int` must match the integer variable `\g__enumext_count_item_ans_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 increase by one (for the `\item` or `\item*` that opens the nest) so that the account remains the same.
c) If there is the option `no-store` we must add the items within this level plus one to maintain the equality.

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.

10.22.1 The check answer mechanism

check-ans
no-store

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

```
1369 \cs_set_protected:Npn \__enumext_tmp:n #1
1370   {
1371     \keys_define:nn { enumext / #1 }
1372       {
1373         check-ans .bool_set:N = \l__enumext_check_ans_bool,
```

```
1374              check-ans  .initial:n  = false,
1375              no-store   .code:n = {
1376                                    \bool_set_false:N \l__enumext_store_ans_bool
1377                                    \bool_set_false:N \l__enumext_check_ans_bool
1378                                 },
1379              no-store  .value_forbidden:n = true,
1380          }
1381      }
1382  \clist_map_inline:nn
1383      {
1384          level-1, level-2, level-3, level-4, enumext*
1385      }
1386      { \__enumext_tmp:n {#1} }
```

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

__enumext_current_env: Function to detect in which environment you are working.

```
1387  \cs_new_protected:Nn \__enumext_current_env:
1388      {
1389          \str_case:en { \@currenvir }
1390              {
1391                  {enumext}
1392                      {
1393                          \int_compare:nNnT { \l__enumext_level_h_int } = { \c_zero_int }
1394                              {
1395                                  \bool_gset_true:N \g__enumext_standar_bool
1396                                  \int_gset:Nn \g__enumext_standar_star_env_int { \inputlineno }
1397                                  \typeout{working-on-enumext}
1398                                  % \int_show:N \g__enumext_standar_star_env_int
1399                              }
1400                      }
1401                  {enumext*}
1402                      {
1403                          \int_compare:nNnT { \l__enumext_level_int } = { \c_zero_int }
1404                              {
1405                                  \bool_gset_true:N \g__enumext_starred_bool
1406                                  \int_gset:Nn \g__enumext_starred_star_env_int { \inputlineno }
1407                                  \typeout{working-on-enumext* ~ \g__enumext_starred_star_env_int}
1408                              }
1409                      }
1410              }
1411      }
```

(*End of definition for* __enumext_current_env:.)

__enumext_check_ans_set_int_level: The function __enumext_check_ans_set_int_level: 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.

```
1412  \cs_new_protected:Nn \__enumext_check_ans_set_int_level:
1413      {
1414          \int_case:nn { \l__enumext_level_int }
1415              {
1416                  { 1 }{
1417                          %\bool_show:N \g__enumext_starred_bool
1418                          \bool_lazy_all:nT
1419                              {
1420                                  { \bool_if_p:N \g__enumext_starred_bool }
1421                                  { \int_compare_p:nNn { \l__enumext_level_h_int } = { \c_one_int } }
1422                              }
1423                              {
1424                                  \int_gdecr:N \g__enumext_count_item_number_int
1425                                  \typeout{ENUMEXT ~ STANDAR ~ NEEEEEEEEEEEESTED}
1426                              }
1427                      }
1428                  { 2 }{
1429                          \int_gdecr:N \g__enumext_count_item_number_int
1430                      }
1431                  { 3 }{
1432                          \int_gdecr:N \g__enumext_count_item_number_int
1433                      }
1434                  { 4 }{
```

```
1435                    \int_gdecr:N \g__enumext_count_item_number_int
1436                  }
1437              }
1438          \int_case:nn { \l__enumext_level_h_int }
1439            {
1440              { 1 }{
1441                    %\bool_show:N \g__enumext_standar_bool
1442                    \bool_if:NT \g__enumext_standar_bool
1443                      {
1444                        \int_gdecr:N \g__enumext_count_item_number_int
1445                        \typeout{ENUMEXT ~ STARRED ~ NEEEEEEEEEEEESTED}
1446                      }
1447                  }
1448              }
1449          }
```

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

10.22.2 Set-up check answer mechanism

`\__enumext_check_ans_count:` The function `\__enumext_check_ans_count:` 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.

```
1450  \cs_new_protected:Nn \__enumext_check_ans_count:
1451    {
1452      \bool_if:NT \l__enumext_check_ans_bool
1453        {
1454          \__enumext_check_ans_set_int_level:
1455        }
1456    }
```

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

`\__enumext_check_ans_active:` The function `\__enumext_check_ans_active:` compare all `\item`'s plus `\item*`'s and `\item`'s with answer for checking answer mechanism and display the appropriate message on the terminal. After the function is executed, we set the integer variables to zero.

```
1457  \cs_new_protected:Nn \__enumext_check_ans_active:
1458    {
1459      \int_compare:nNnTF
1460        { \g__enumext_count_item_number_int } = { \g__enumext_count_item_anskey_int }
1461        {
1462          \msg_term:nnV { enumext } { items-same-answer } \g__enumext_store_name_tl
1463        }
1464        {
1465          \msg_warning:nnV { enumext } { item-different-answer } \g__enumext_store_name_tl
1466        }
1467      \int_gzero:N \g__enumext_count_item_number_int
1468      \int_gzero:N \g__enumext_count_item_anskey_int
1469    }
```

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

10.23 Keys and functions associated with storage

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

```
1470  \cs_set_protected:Npn \__enumext_tmp:n #1
1471    {
1472      \keys_define:nn { enumext / #1 }
1473        {
1474          wrap-ans   .cs_set_protected:Np = \__enumext_anskey_wrapper:n ##1,
1475          wrap-ans   .initial:n  = \fbox{##1},
1476          wrap-ans   .value_required:n = true,
1477          mark-ans   .code:n     = \tl_set:Nn \l__enumext_mark_answer_sym_tl {##1},
1478          mark-ans   .initial:n  = \textasteriskcentered,
1479          mark-ans   .value_required:n = true,
1480          mark-pos   .choice:,
1481          mark-pos  / left   .code:n = \str_set:Nn \l__enumext_mark_position_str { l },
1482          mark-pos  / right  .code:n = \str_set:Nn \l__enumext_mark_position_str { r },
1483          mark-pos   .initial:n     = right,
```

```
1484        mark-pos   .value_required:n = true,
1485        show-ans   .code:n     = \bool_set_true:N  \l__enumext_show_answer_bool
1486                                \bool_set_false:N \l__enumext_show_position_bool,
1487        show-ans   .value_forbidden:n = true,
1488        show-pos   .code:n     = \bool_set_true:N \l__enumext_show_position_bool
1489                                \bool_set_false:N  \l__enumext_show_answer_bool,
1490        show-pos   .value_forbidden:n = true,
1491        mark-ref   .code:n     = \tl_set:Nn \l__enumext_mark_ref_sym_tl {##1},
1492        mark-ref   .initial:n  = \textasteriskcentered,
1493        mark-ref   .value_required:n = true,
1494        store-ref  .bool_set:N = \l__enumext_store_ref_key_bool,
1495        store-ref  .initial:n  = false,
1496      }
1497    }
1498 \clist_map_inline:nn { level-1, enumext* } { \__enumext_tmp:n {#1} }
```

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

mark-pos
show-ans

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

```
1499 \cs_set_protected:Npn \__enumext_tmp:n #1
1500   {
1501     \keys_define:nn { enumext / #1 }
1502       {
1503         mark-pos .choice:,
1504         mark-pos / left   .code:n    = \str_set:Nn \l__enumext_mark_position_str { l },
1505         mark-pos / right  .code:n    = \str_set:Nn \l__enumext_mark_position_str { r },
1506         mark-pos         .initial:n = right,
1507         mark-pos .value_required:n  = true,
1508         show-ans .code:n   = \bool_set_true:N  \l__enumext_show_answer_bool
1509                             \bool_set_false:N \l__enumext_show_position_bool,
1510         show-ans .value_forbidden:n = true,
1511         show-pos .code:n   = \bool_set_true:N  \l__enumext_show_position_bool
1512                             \bool_set_false:N \l__enumext_show_answer_bool,
1513         show-pos .value_forbidden:n = true,
1514       }
1515   }
1516 \clist_map_inline:nn { keyans, keyans* } { \__enumext_tmp:n {#1} }
```

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

columns*
columns-sep*

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

```
1517 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1518   {
1519     \keys_define:nn { enumext / #1 }
1520       {
1521         columns*     .code:n = \bool_set_true:c { l__enumext_store_columns_#2_bool }
1522                                \int_set:cn { l__enumext_store_columns_#2_int } {##1}
1523                                \tl_put_right:ce { l__enumext_store_opt_#2_tl }
1524                                  {
1525                                    columns = \exp_not:v { l__enumext_store_columns_#2_int },
1526                                  },,
1527         columns*     .value_required:n  = true,
1528         columns-sep* .code:n = \bool_set_true:c { l__enumext_store_columns_sep_#2_bool }
1529                                \dim_set:cn { l__enumext_store_columns_sep_#2_dim } {##1}
1530                                \tl_put_right:ce { l__enumext_store_opt_#2_tl }
1531                                  {
1532                                    columns-sep = \exp_not:v { l__enumext_store_columns_sep_#2_di
1533                                  },
1534         columns-sep* .value_required:n  = true,
1535       }
1536   }
1537 \clist_map_inline:nn
1538   {
1539     {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {enumext*}{vii}
1540   }
1541   { \__enumext_tmp:nn #1 }
```

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

10.23.1 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.

```
1542 \cs_generate_variant:Nn \prop_gput_if_not_in:Nnn { cen }
1543 \cs_new_protected:Npn \__enumext_store_addto_prop:n #1
1544   {
1545     \prop_gput_if_not_in:cen { g__enumext_ \l__enumext_store_name_tl _prop }
1546       {
1547         \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } + 1 }
1548       }
1549       { #1 }
1550   }
1551 \cs_generate_variant:Nn \__enumext_store_addto_prop:n { V }
```

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

10.23.2 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.

```
1552 \cs_new_protected:Npn \__enumext_store_addto_seq:n #1
1553   {
1554     \seq_gput_right:cn { g__enumext_ \l__enumext_store_name_tl _seq } { #1 }
1555   }
1556 \cs_generate_variant:Nn \__enumext_store_addto_seq:n { v, V }
```

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

10.23.3 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`.

```
1557 \cs_new_protected:Nn \__enumext_store_level_open:
1558   {
1559     \bool_if:NT \l__enumext_store_ans_bool
1560       {
1561         \tl_if_empty:cTF { l__enumext_store_opt_ \__enumext_level: _tl }
1562           {
1563             \__enumext_store_addto_seq:n
1564               {
1565                 \item \begin{enumext}
1566               }
1567           }
1568           {
1569             \tl_put_left:cn { l__enumext_store_opt_ \__enumext_level: _tl }
1570               {
1571                 \item \begin{enumext} [
1572               }
1573             \tl_put_right:cn { l__enumext_store_opt_ \__enumext_level: _tl }
1574               {
1575                 ]
1576               }
1577             \__enumext_store_addto_seq:v { l__enumext_store_opt_ \__enumext_level: _tl }
1578           }
1579       }
1580   }
1581 \cs_new_protected:Nn \__enumext_store_level_close:
1582   {
1583     \bool_if:NT \l__enumext_store_ans_bool
1584       {
```

```
1585        \__enumext_store_addto_seq:n { \end{enumext} }
1586      }
1587    }
```

(*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.

```
1588 \cs_new_protected:Nn \__enumext_store_level_open_vii:
1589    {
1590      \bool_if:NT \l__enumext_store_ans_bool
1591        {
1592          \tl_if_empty:NTF \l__enumext_store_opt_vii_tl
1593            {
1594              \__enumext_store_addto_seq:n
1595                {
1596                  \item \mode_leave_vertical:
1597                    \vspace { -\skip_eval:n { \baselineskip + \parsep } }
1598                    \begin{enumext*}[before={\setlength{\topsep}{0pt}},]
1599                }
1600            }
1601            {
1602              \tl_put_left:Nn \l__enumext_store_opt_vii_tl
1603                {
1604                  \item \mode_leave_vertical:
1605                    \vspace { -\skip_eval:n { \baselineskip + \parsep } }
1606                    \begin{enumext*}[before={\setlength{\topsep}{0pt}},
1607                }
1608              \tl_put_right:Nn \l__enumext_store_opt_vii_tl
1609                {
1610                  ]
1611                }
1612              \__enumext_store_addto_seq:V \l__enumext_store_opt_vii_tl
1613            }
1614        }
1615    }
1616 \cs_new_protected:Nn \__enumext_store_level_close_vii:
1617    {
1618      \bool_if:NT \l__enumext_store_ans_bool
1619        {
1620          \__enumext_store_addto_seq:n { \end{enumext*} }
1621        }
1622    }
```

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

10.23.4 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`

```
1623 \cs_new_protected:Nn \__enumext_print_keyans_box:NN
1624    {
1625      \mode_leave_vertical:
1626      \skip_horizontal:n { -\dim_use:N #2 }
1627      \makebox[0pt][ r ]
1628        {
1629          \makebox[ \dim_use:N #1 ][ \l__enumext_mark_position_str ]
1630            {
1631              \tl_use:N \l__enumext_mark_answer_sym_tl
1632            }
1633        }
1634      \skip_horizontal:n { \dim_use:N #2 }
1635    }
1636 \cs_generate_variant:Nn \__enumext_print_keyans_box:NN { cc }
```

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

10.24 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.

```
1637 \keys_define:nn { enumext / anskey }
1638   {
1639     item-sym* .tl_set:N   = \l__enumext_store_item_symbol_tl,
1640     item-sym* .value_required:n = true,
1641     item-pos* .dim_set:N  = \l__enumext_store_item_symbol_sep_dim,
1642     item-pos* .value_required:n = true,
1643     store-brk .bool_set:N = \l__enumext_store_columns_break_bool,
1644     store-brk .default:n  = true,
1645     store-brk .value_forbidden:n = true,
1646   }
```

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*⟩.

```
1647 \NewDocumentCommand \anskey { d() s o +m }
1648   {
1649     \bool_if:NF \l__enumext_store_active_bool
1650       {
1651         \msg_error:nnnn { enumext } { anskey-wrong-place }{ anskey }{ enumext }
1652       }
1653     \int_compare:nNnT { \l__enumext_keyans_level_int } = { 1 }
1654       {
1655         \msg_error:nnnn { enumext } { command-wrong-place }{ anskey }{ keyans }
1656       }
1657     \int_compare:nNnT { \l__enumext_keyans_pic_level_int } = { 1 }
1658       {
1659         \msg_error:nnnn { enumext } { command-wrong-place }{ anskey }{ keyanspic }
1660       }
1661     \group_begin:
1662       \bool_if:NT \l__enumext_store_ans_bool
1663         {
1664           \bool_if:NT \l__enumext_check_ans_bool
1665             {
1666               \int_gincr:N \g__enumext_count_item_anskey_int
1667             }
1668           \__enumext_store_anskey_code:nnnn {#1} {#2} {#3} {#4}
1669         }
1670     \group_end:
1671   }
```

(*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 store-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.

```
1672 \cs_new_protected:Npn \__enumext_store_anskey_code:nnnn #1 #2 #3 #4
1673   {
1674     \__enumext_store_addto_prop:n {#4}
1675     \bool_if:NT \l__enumext_store_ref_key_bool
1676       {
1677         \__enumext_store_internal_ref:
1678       }
1679     \__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.

```
1680        \tl_clear:N \l__enumext_store_anskey_arg_tl
1681        \tl_if_novalue:nF {#3}
1682          {
1683            \keys_set:nn { enumext / anskey } {#3}
1684          }
1685        \bool_lazy_and:nnT
1686          { \bool_if_p:N \l__enumext_store_columns_break_bool }
1687          { \bool_not_p:n { \l__enumext_starred_bool } }
1688          {
1689            \tl_put_left:Nn \l__enumext_store_anskey_arg_tl { \columnbreak }
1690          }
1691        \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).

```
1692        \tl_if_novalue:nF {#1}
1693          {
1694            \int_set:Nn \l__enumext_store_columns_join_int {#1}
1695            \bool_if:NT \l__enumext_starred_bool
1696              {
1697                \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
1698                  {
1699                    ( \exp_not:V \l__enumext_store_columns_join_int )
1700                  }
1701              }
1702          }
```

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.

```
1703        \bool_if:nTF {#2}
1704          {
1705            \tl_put_right:Nn \l__enumext_store_anskey_arg_tl { * }
1706            \tl_if_empty:NF \l__enumext_store_item_symbol_tl
1707              {
1708                \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
1709                  {
1710                    [ \exp_not:V \l__enumext_store_item_symbol_tl ]
1711                  }
1712              }
1713            \dim_compare:nT
1714              {
1715                \l__enumext_store_item_symbol_sep_dim != \c_zero_dim
1716              }
1717              {
1718                \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
1719                  {
1720                    [ \exp_not:V \l__enumext_store_item_symbol_sep_dim ]
1721                  }
1722              }
1723            \tl_put_right:Nn \l__enumext_store_anskey_arg_tl {#4}
1724          }
1725          {
1726            \tl_put_right:Nn \l__enumext_store_anskey_arg_tl {#4}
1727          }
```

Finally we check if the store-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*⟩.

```
1728        \bool_lazy_and:nnT
1729          { \bool_if_p:N \l__enumext_store_ref_key_bool }
1730          { \bool_if_p:N \l__enumext_hyperref_bool }
1731          {
1732            \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
1733              {
1734                \hfill \exp_not:N \hyperlink { \exp_not:V \l__enumext_newlabel_arg_one_tl }
1735                  { \exp_not:V \l__enumext_mark_ref_sym_tl }
1736              }
1737          }
1738        \__enumext_store_addto_seq:V \l__enumext_store_anskey_arg_tl
1739      }
```

(*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 `store-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.

```
1740  \cs_new_protected:Nn \__enumext_store_internal_ref:
1741    {
1742      \cs_set_protected:Npn \__enumext_tmp:n ##1
1743        {
1744          \tl_set_eq:cc { l__enumext_label_copy_##1_tl } { l__enumext_label_##1_tl }
1745          \tl_reverse:c { l__enumext_label_copy_##1_tl }
1746          \tl_remove_once:cn { l__enumext_label_copy_##1_tl } { . }
1747          \tl_reverse:c { l__enumext_label_copy_##1_tl }
1748        }
1749      \clist_map_inline:nn { i, ii, iii, iv, vii } { \__enumext_tmp:n {##1} }
1750      \cs_set:Npn \__enumext_tmp:n ##1
1751        { . \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.

```
1752      \bool_lazy_all:nT
1753        {
1754          { \bool_if_p:N \g__enumext_starred_bool }
1755          { \int_compare_p:nNn { \l__enumext_level_int } = { \c_zero_int } }
1756        }
1757        {
1758          \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
1759            { \tl_use:N \l__enumext_label_copy_vii_tl }
1760        }
1761      \bool_lazy_all:nT
1762        {
1763          { \bool_if_p:N \l__enumext_standar_bool }
1764          { \bool_if_p:N \g__enumext_starred_bool }
1765          { \int_compare_p:nNn { \l__enumext_level_int } > { \c_zero_int } }
1766        }
1767        {
1768          \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
1769            {
1770              \tl_use:N \l__enumext_label_copy_vii_tl
1771              \int_step_function:nnN { 1 } { \l__enumext_level_int } \__enumext_tmp:n
1772            }
1773        }
```

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.

```
1774      \bool_lazy_all:nT
1775        {
1776          { \bool_if_p:N \l__enumext_standar_bool }
1777          { \int_compare_p:nNn { \l__enumext_level_int } > { \c_zero_int } }
1778          { \int_compare_p:nNn { \l__enumext_level_h_int } = { \c_zero_int } }
1779          { \bool_not_p:n { \l__enumext_starred_bool } }
1780        }
1781        {
1782          \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
1783            {
1784              \tl_use:N \l__enumext_label_copy_i_tl
1785              \int_step_function:nnN { 2 } { \l__enumext_level_int } \__enumext_tmp:n
1786            }
1787        }
1788      \cs_set:Npn \__enumext_tmp:n ##1
1789        { \tl_use:c { l__enumext_label_copy_ \int_to_roman:n {##1} _tl } }
1790      \bool_lazy_all:nT
1791        {
1792          { \bool_if_p:N \l__enumext_standar_bool }
1793          { \int_compare_p:nNn { \l__enumext_level_int } > { \c_zero_int } }
1794          { \bool_not_p:n { \g__enumext_starred_bool } }
1795          { \int_compare_p:nNn { \l__enumext_level_h_int } > { \c_zero_int } }
1796        }
1797        {
1798          \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
1799            {
```

```
1800              \int_step_function:nnN { 1 } { \l__enumext_level_int } \__enumext_tmp:n
1801                . \tl_use:N \l__enumext_label_copy_vii_tl
1802            }
1803        }
```

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

```
1804        \tl_put_right:Ne \l__enumext_newlabel_arg_one_tl
1805          {
1806            \l__enumext_store_name_tl \c_colon_str
1807            \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } }
1808          }
```

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.

```
1809        \tl_put_right:Ne \l__enumext_store_write_aux_file_tl
1810          {
1811            \__enumext_newlabel:nn
1812              { \exp_not:V \l__enumext_newlabel_arg_one_tl }
1813              { \l__enumext_newlabel_arg_two_tl }
1814          }
1815        \l__enumext_store_write_aux_file_tl
1816      }
```

(*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.

```
1817 \cs_new_protected:Npn \__enumext_store_anskey_show_wrap:n #1
1818   {
1819     \par
1820     \bool_if:NT \l__enumext_starred_bool
1821       {
1822         \cs_set:Nn \__enumext_level: { vii }
1823       }
1824     \__enumext_print_keyans_box:cc
1825       { l__enumext_labelwidth_ \__enumext_level: _dim }
1826       { l__enumext_labelsep_ \__enumext_level: _dim }
1827     \__enumext_anskey_wrapper:n { #1 }
1828   }
```

(*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.

```
1829 \cs_new_protected:Npn \__enumext_store_anskey_show_left:n #1
1830   {
1831     \bool_if:NT \l__enumext_show_answer_bool
1832       {
1833         \__enumext_store_anskey_show_wrap:n { #1 }
1834       }
1835     \bool_if:NT \l__enumext_show_position_bool
1836       {
1837         \tl_set:Ne \l__enumext_mark_answer_sym_tl
1838           {
1839             \group_begin:
1840             \exp_not:N \normalfont
1841             \exp_not:N \footnotesize [ \int_eval:n
1842               {
1843                 \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
1844               }
1845             ]
1846             \group_end:
1847           }
1848         \__enumext_store_anskey_show_wrap:n { #1 }
1849       }
1850   }
```

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

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

10.25.1   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`.

```
1851 \cs_new_protected:Npn \__enumext_keyans_addto_prop:n #1
1852   {
1853     \tl_clear:N \l__enumext_store_keyans_label_tl
1854     \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
1855       {
1856         \tl_put_right:Ne \l__enumext_store_keyans_label_tl { \l__enumext_label_vi_tl }
1857       }
1858       {
1859         \tl_put_right:Ne \l__enumext_store_keyans_label_tl { \l__enumext_label_v_tl }
1860       }
1861     \tl_if_novalue:nF { #1 }
1862       {
1863         \tl_put_right:Ne \l__enumext_store_keyans_label_tl { \c_space_tl #1 }
1864       }
1865     \__enumext_store_addto_prop:V \l__enumext_store_keyans_label_tl
1866   }
```

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

10.25.2   The `store-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 `store-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.

```
1867 \cs_new_protected:Nn \__enumext_keyans_store_ref:
1868   {
1869     \bool_if:NT \l__enumext_store_ref_key_bool
1870       {
1871         \cs_set_protected:Npn \__enumext_tmp:n ##1
1872           {
1873             \tl_set_eq:cc { l__enumext_label_copy_##1_tl } { l__enumext_label_##1_tl }
1874             \tl_reverse:c { l__enumext_label_copy_##1_tl }
1875             \tl_remove_once:cn { l__enumext_label_copy_##1_tl } { . }
1876             \tl_reverse:c { l__enumext_label_copy_##1_tl }
1877           }
1878         \clist_map_inline:nn { i, v, vi, vii, viii } { \__enumext_tmp:n {##1} }
1879         \__enumext_keyans_store_ref_aux_i:
1880       }
1881   }
```

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`.

```
1882 \cs_new_protected:Nn \__enumext_keyans_store_ref_aux_i:
1883   {
1884     \bool_if:NT \g__enumext_starred_bool
1885       {
1886         \tl_set_eq:NN \l__enumext_label_copy_i_tl \l__enumext_label_copy_vii_tl
1887       }
1888     \int_compare:nNnT { \l__enumext_keyans_pic_level_int } = { 1 }
1889       {
1890         \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
1891           { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_vi_tl }
1892       }
1893     \int_compare:nNnT { \l__enumext_keyans_level_int } = { 1 }
1894       {
1895         \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
```

```
1896              { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_v_tl }
1897          }
1898        \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
1899          {
1900            \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
1901              { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_viii_tl }
1902          }
1903        \tl_put_right:Ne \l__enumext_newlabel_arg_one_tl
1904          {
1905            \l__enumext_store_name_tl \c_colon_str
1906            \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } }
1907          }
1908        \__enumext_keyans_store_ref_aux_ii:
1909      }
```

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.

```
1910  \cs_new_protected:Nn \__enumext_keyans_store_ref_aux_ii:
1911      {
1912        \tl_put_right:Ne \l__enumext_store_write_aux_file_tl
1913          {
1914            \__enumext_newlabel:nn
1915              { \exp_not:V \l__enumext_newlabel_arg_one_tl }
1916              { \l__enumext_newlabel_arg_two_tl }
1917          }
1918        \l__enumext_store_write_aux_file_tl
1919      }
```

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

10.25.3   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.

```
1920  \cs_new_protected:Npn \__enumext_keyans_addto_seq:n #1
1921      {
1922        \tl_clear:N \l__enumext_store_keyans_label_tl
1923        \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
1924          {
1925            \tl_put_right:Ne \l__enumext_store_keyans_label_tl { \item \l__enumext_label_vi_tl }
1926          }
1927          {
1928            \tl_put_right:Ne \l__enumext_store_keyans_label_tl { \item \l__enumext_label_v_tl }
1929          }
1930        \tl_if_novalue:nF { #1 }
1931          {
1932            \tl_put_right:Ne \l__enumext_store_keyans_label_tl { \c_space_tl #1 }
1933          }
1934        \__enumext_keyans_addto_seq_link:
1935      }
```

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

```
1936  \cs_new_protected:Nn \__enumext_keyans_addto_seq_link:
1937      {
1938        \bool_lazy_and:nnT
1939          { \bool_if_p:N \l__enumext_store_ref_key_bool }
1940          { \bool_if_p:N \l__enumext_hyperref_bool }
1941          {
1942            \tl_put_right:Ne \l__enumext_store_keyans_label_tl
1943              {
1944                \hfill \exp_not:N \hyperlink
1945                  {
1946                    \exp_not:V \l__enumext_newlabel_arg_one_tl
```

```
1947              }
1948                { \exp_not:V \l__enumext_mark_ref_sym_tl }
1949              }
1950          }
1951      \__enumext_store_addto_seq:V \l__enumext_store_keyans_label_tl
1952      \tl_gset:NV \g__enumext_check_ans_item_tl \l__enumext_store_keyans_label_tl
1953      \bool_if:NT \l__enumext_check_ans_bool
1954          {
1955              \int_gincr:N \g__enumext_count_item_anskey_int
1956          }
1957      }
```

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

10.25.4 Check for starred commands

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

```
1958  \cs_new_protected:Npn \__enumext_keyans_check_ans:nn #1 #2
1959      {
1960      \tl_if_empty:NTF \g__enumext_check_ans_item_tl
1961          {
1962              \msg_warning:nnnn { enumext } { missing-starred }{ #1 }{ #2 }
1963          }
1964          { \tl_gclear:N \g__enumext_check_ans_item_tl }
1965      }
```

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

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

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

__enumext_keyans_show_left:n Common function to show *starred commands* \item* and ⟨position⟩ of stored content in ⟨prop list⟩ for keyans and keyanspic. Need add 1 to \l__enumext_show_position_bool for keyans environment.

```
1966  \cs_new_protected:Npn \__enumext_keyans_show_left:n #1
1967      {
1968      \bool_if:NT \l__enumext_show_answer_bool
1969          {
1970              \tl_put_left:Nn \l__enumext_label_v_tl
1971                  {
1972                      \__enumext_print_keyans_box:NN
1973                          \l__enumext_labelwidth_i_dim
1974                          \l__enumext_labelsep_i_dim
1975                  }
1976              \tl_if_novalue:nF { #1 }
1977                  { \tl_put_right:Nn \l__enumext_label_v_tl { \c_space_tl [ #1 ] } }
1978          }
1979      \bool_if:NT \l__enumext_show_position_bool
1980          {
1981              \tl_set:Ne \l__enumext_mark_answer_sym_tl
1982                  {
1983                      \group_begin:
1984                      \exp_not:N \normalfont
1985                      \exp_not:N \footnotesize [ \int_eval:n
1986                          {
1987                              \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } + \c_one_int
1988                          }
1989                      ]
1990                      \group_end:
1991                  }
1992              \tl_put_left:Nn \l__enumext_label_v_tl
1993                  {
1994                      \__enumext_print_keyans_box:NN
1995                          \l__enumext_labelwidth_i_dim
1996                          \l__enumext_labelsep_i_dim
1997                  }
1998              \tl_if_novalue:nF { #1 }
1999                  { \tl_put_right:Nn \l__enumext_label_v_tl { \c_space_tl [ #1 ] } }
2000          }
2001      }
```

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

10.26 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*`.

```
2002 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
2003    {
2004      \keys_define:nn { enumext / #1 }
2005        {
2006          item-sym* .tl_set:c  = { l__enumext_item_symbol_#2_tl },
2007          item-sym* .value_required:n = true,
2008          item-sym* .initial:n  = {$\star$},
2009          item-pos* .dim_set:c  = { l__enumext_item_symbol_sep_#2_dim },
2010          item-pos* .value_required:n = true,
2011        }
2012    }
2013 \clist_map_inline:nn
2014    {
2015      {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {enumext*}{vii}
2016    }
2017    { \__enumext_tmp:nn #1 }
```

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

10.27 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.

```
2018 \cs_new_protected:Nn \__enumext_footnotetext:nn
2019    {
2020      \footnotetext[#1]{#2}
2021    }
2022 \cs_new_protected:Nn \__enumext_renew_footnote:
2023    {
2024      \seq_gclear:N \g__enumext_footnote_arg_seq
2025      \seq_gclear:N \g__enumext_footnote_int_seq
2026      \RenewDocumentCommand \footnote { o +m }
2027        {
2028          \tl_if_novalue:nTF {##1}
2029            {
2030              \stepcounter{footnote}
2031              \int_gset_eq:Nc \g__enumext_footnote_int { c@footnote }
2032            }
2033            {
2034              \int_gset:Nn \g__enumext_footnote_int { ##1 }
2035            }
2036          \footnotemark [ \g__enumext_footnote_int ]
2037          \seq_gput_right:Nn \g__enumext_footnote_arg_seq { ##2 }
2038          \seq_gput_right:NV \g__enumext_footnote_int_seq \g__enumext_footnote_int
2039        }
2040    }
2041 \cs_new_protected:Nn \__enumext_print_footnote:
2042    {
2043      \seq_if_empty:NF \g__enumext_footnote_int_seq
2044        {
2045          \seq_map_pairwise_function:NNN
2046            \g__enumext_footnote_int_seq
2047            \g__enumext_footnote_arg_seq
2048            \__enumext_footnotetext:nn
2049        }
2050    }
```

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

10.28 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.

10.28.1 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: (§10.29).

```
2051  \cs_new_protected:Npn \__enumext_default_item:n #1
2052    {
2053      \tl_if_novalue:nTF {#1}
2054        {
2055          \bool_if:NT \l__enumext_check_ans_bool
2056            {
2057              \int_gincr:N \g__enumext_count_item_number_int
2058            }
2059          \bool_set_true:c { l__enumext_wrap_label_ \__enumext_level: _bool }
2060          \__enumext_item_std:w \tl_use:c { l__enumext_fake_item_indent_ \__enumext_level: _tl }
2061        }
2062        {
2063          \bool_set_eq:cc
2064            { l__enumext_wrap_label_ \__enumext_level: _bool }
2065            { l__enumext_wrap_label_opt_ \__enumext_level: _bool }
2066          \__enumext_item_std:w [#1] \tl_use:c { l__enumext_fake_item_indent_ \__enumext_level: _tl
2067        }
2068    }
```

(*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: (§10.29).

```
2069  \cs_new_protected:Npn \__enumext_starred_item:nn #1 #2
2070    {
2071      \tl_if_novalue:nF {#1}
2072        {
2073          \tl_set:cn { l__enumext_item_symbol_ \__enumext_level: _tl } {#1}
2074        }
2075      \tl_gset_eq:Nc \g__enumext_item_symbol_tl { l__enumext_item_symbol_ \__enumext_level: _tl }
2076      \tl_if_novalue:nTF {#2}
2077        {
2078          \dim_set_eq:cc
2079            { l__enumext_item_symbol_sep_ \__enumext_level: _dim }
2080            { l__enumext_labelsep_ \__enumext_level: _dim }
2081        }
2082        {
2083          \dim_set:cn { l__enumext_item_symbol_sep_ \__enumext_level: _dim } {#2}
2084        }
2085      \bool_if:NT \l__enumext_check_ans_bool
2086        {
2087          \int_gincr:N \g__enumext_count_item_number_int
```

```
2088          }
2089        \bool_set_true:c { l__enumext_wrap_label_ \__enumext_level: _bool }
2090        \__enumext_item_std:w \tl_use:c { l__enumext_fake_item_indent_ \__enumext_level: _tl }
2091      }
```

(*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).

```
2092  \cs_new_protected:Nn \__enumext_redefine_item:
2093    {
2094      \RenewDocumentCommand \item { s o o }
2095        {
2096          \bool_if:nTF {##1}
2097            {
2098              \__enumext_starred_item:nn {##2} {##3}
2099            }
2100            { \__enumext_default_item:n {##2} }
2101        }
2102    }
```

(*End of definition for* __enumext_redefine_item:.)

10.28.2   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.

```
2103  \cs_new_protected:Npn \__enumext_keyans_default_item:n #1
2104    {
2105      \tl_if_novalue:nTF { #1 }
2106        {
2107          \bool_set_true:N \l__enumext_wrap_label_v_bool
2108          \__enumext_item_std:w \tl_use:N \l__enumext_fake_item_indent_v_tl
2109        }
2110        {
2111          \bool_set_eq:NN \l__enumext_wrap_label_v_bool \l__enumext_wrap_label_opt_v_bool
2112          \__enumext_item_std:w [#1] \tl_use:N \l__enumext_fake_item_indent_v_tl
2113        }
2114    }
```

(*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⟩.

```
2115  \cs_new_protected:Npn \__enumext_keyans_starred_item:n #1
2116    {
2117      \tl_set_eq:NN \l__enumext_keyans_tmpa_tl \l__enumext_label_v_tl
2118      \__enumext_keyans_show_left:n { #1 }
2119      \bool_set_true:N \l__enumext_wrap_label_v_bool
2120      \__enumext_item_std:w \tl_use:N \l__enumext_fake_item_indent_v_tl
```

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 store-ref key is active and finally *store* it in the ⟨sequence⟩.

```
2121      \tl_set_eq:NN \l__enumext_label_v_tl \l__enumext_keyans_tmpa_tl
2122      \__enumext_keyans_addto_prop:n { #1 }
2123      \__enumext_keyans_store_ref:
2124      \__enumext_keyans_addto_seq:n { #1 }
2125    }
```

(*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).

```
2126  \cs_new_protected:Nn \__enumext_keyans_redefine_item:
2127    {
2128      \RenewDocumentCommand \item { s o }
2129        {
2130          \bool_if:nTF {##1}
2131            {
2132              \peek_remove_spaces:n
2133                {
2134                  \__enumext_keyans_starred_item:n {##2}
2135                }
2136            }
2137            {
2138              \__enumext_keyans_default_item:n {##2}
2139            }
2140        }
2141    }
```

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

10.29 Redefining \makelabel command

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

10.29.1 Redefining \makelabel for enumext

__enumext_item_starred:

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

```
2142  \cs_new_protected:Nn \__enumext_item_starred:
2143    {
2144      \tl_if_empty:cF { l__enumext_item_symbol_ \__enumext_level: _tl }
2145        {
2146          \mode_leave_vertical:
2147          \skip_horizontal:n { -\dim_use:c { l__enumext_item_symbol_sep_ \__enumext_level: _dim } }
2148          \makebox[ 0pt ][ r ]{ \g__enumext_item_symbol_tl }
2149          \skip_horizontal:n { \dim_use:c { l__enumext_item_symbol_sep_ \__enumext_level: _dim } }
2150        }
2151    }
```

(*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).

```
2152  \cs_new_protected:Nn \__enumext_make_label:
2153    {
2154      \RenewDocumentCommand \makelabel { m }
2155        {
2156          \tl_use:c { l__enumext_label_fill_left_ \__enumext_level: _tl }
2157          \tl_use:c { l__enumext_label_font_style_ \__enumext_level: _tl }
2158          \bool_if:cTF { l__enumext_wrap_label_ \__enumext_level: _bool }
2159            {
2160              \__enumext_item_starred:
2161              \use:c { __enumext_wrapper_label_ \__enumext_level: :n } { ##1 }
2162            }
2163            { ##1 }
2164          \tl_use:c { l__enumext_label_fill_right_ \__enumext_level: _tl }
2165          \tl_gclear:N \g__enumext_item_symbol_tl
2166        }
2167    }
```

(*End of definition for* __enumext_make_label:.)

10.29.2 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).

```
2168  \cs_new_protected:Nn \__enumext_keyans_make_label:
2169    {
2170      \RenewDocumentCommand \makelabel { m }
2171        {
2172          \tl_use:N \l__enumext_label_fill_left_v_tl
2173          \tl_use:N \l__enumext_label_font_style_v_tl
2174          \bool_if:NTF \l__enumext_wrap_label_v_bool
2175            {
2176              \__enumext_wrapper_label_v:n { ##1 }
2177            }
2178            { ##1 }
2179          \tl_use:N \l__enumext_label_fill_right_v_tl
2180        }
2181    }
```

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

10.30 Calculation of \leftmargin and \itemindent

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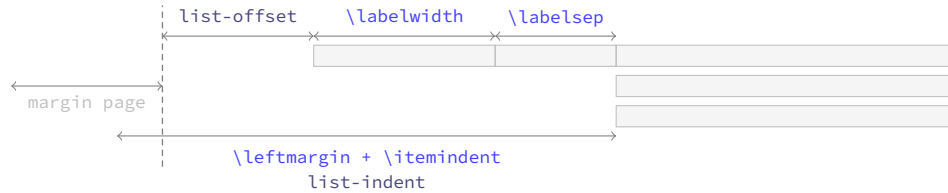
Figure 9: Representation of standard horizontal lengths in list environment.

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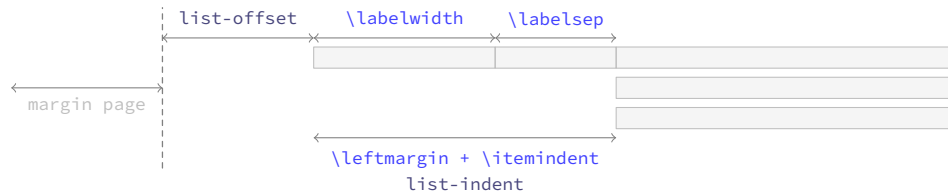
Figure 10: Representation of horizontal lengths concept in list in enumext.

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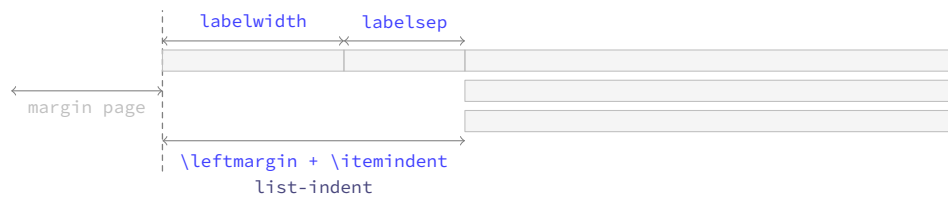
Figure 11: Default horizontal lengths in enumext.

__enumext_calc_hspace:NNNNNNN
__enumext_calc_hspace:ccccccc

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

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

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

This function is passed to __enumext_list_arg_two_X: which is used in the definition of the enumext and keyans environments (§10.31).

```
2182  \cs_new_protected:Npn \__enumext_calc_hspace:NNNNNNN #1 #2 #3 #4 #5 #6 #7
2183    {
2184      \dim_compare:nNnT { #1 } < { \c_zero_dim }
```

```
2185          {
2186              \msg_warning:nnnV { enumext } { width-non-positive }{ labelwidth }{ #1 }
2187              \dim_set:Nn #1 { \dim_abs:n { #1 } }
2188          }
2189        \dim_compare:nNnT { #2 } < { \c_zero_dim }
2190          {
2191              \msg_warning:nnnV { enumext } { width-negative }{ labelsep }{ #2 }
2192              \dim_set:Nn #2 { \dim_abs:n { #2 } }
2193          }
```

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

```
2194        \bool_if:nF #7 { \dim_set:Nn #4 { #1 + #2} }
```

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

```
2195        \dim_compare:nNnTF { #4 } < { \c_zero_dim }
2196          {
2197              \dim_set:Nn #6 { #1 + #2 - #4}
2198              \dim_set:Nn #5 { #1 + #2 + #3 - #6 }
2199          }
2200          {
2201              \dim_compare:nNnT { #4 } = { #1 + #2 }
2202                { \dim_set:Nn #6 { \c_zero_dim } }
2203              \dim_compare:nNnT { #4 } < { #1 + #2 }
2204                { \dim_set:Nn #6 { #1 + #2 - #4} }
2205              \dim_compare:nNnT { #4 } > { #1 + #2 }
2206                {
2207                  \dim_set:Nn #6 { -#1 - #2 + #4}
2208                  \dim_set:Nn #6 { #6*-1}
2209                }
2210              \dim_set:Nn #5 { #1 + #2 + #3 - #6 }
2211          }
2212      }
2213  \cs_generate_variant:Nn \__enumext_calc_hspace:NNNNNNN { ccccccc }
```

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

10.31   Setting second argument of the lists

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

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

In this function for the second list argument we will implement the keys `start`, `resume` and `show-length` together with the redefinition of `\item` for `enumext` and `keyans` environments. We will "not set" `\leftmargini`, `\leftmarginii`, `\leftmarginiii` or `\leftmarginiv`, in this case, we will directly set the parameters for vertical and horizontal list spacing per level.

```
2214  \cs_set_protected:Npn \__enumext_tmp:n #1
2215    {
2216      \cs_new_protected:cpn { __enumext_list_arg_two_#1: }
2217        {
2218          \__enumext_calc_hspace:ccccccc
2219            { l__enumext_labelwidth_#1_dim } { l__enumext_labelsep_#1_dim }
2220            { l__enumext_listoffset_#1_dim } { l__enumext_leftmargin_tmp_#1_dim }
2221            { l__enumext_leftmargin_#1_dim } { l__enumext_itemindent_#1_dim }
2222            { l__enumext_leftmargin_tmp_#1_bool }
2223          \clist_map_inline:nn
2224            { labelsep, labelwidth, itemindent, leftmargin, rightmargin, listparindent }
2225            { \dim_set_eq:cc {####1} { l__enumext_####1_#1_dim } }
2226          \clist_map_inline:nn { topsep, parsep, partopsep, itemsep }
2227            { \skip_set_eq:cc {####1} { l__enumext_####1_#1_skip } }
2228          \usecounter { enumX#1 }
2229          \bool_lazy_and:nnTF
2230            { \str_if_eq_p:nn {#1} { i } }
2231            { \bool_if_p:N \l__enumext_resume_bool }
2232            { \setcounter { enumXi  } { \int_eval:n { \g__enumext_resume_int } } }
2233            {
2234              \setcounter { enumX#1 }
2235                { \int_eval:n { \int_use:c { l__enumext_start_#1_int } - 1 } }
2236            }
2237          \str_if_eq:nnTF {#1} { v }
```

```
2238                    {
2239                      \__enumext_keyans_redefine_item:
2240                      \__enumext_keyans_make_label:
2241                      \__enumext_keyans_fake_item:
2242                      \bool_if:cT { l__enumext_show_length_#1_bool }
2243                        {
2244                          \msg_term:nnnn { enumext } { list-lengths-not-nested } { v } { keyans }
2245                        }
2246                    }
2247                    {
2248                      \__enumext_redefine_item:
2249                      \__enumext_make_label:
2250                      \__enumext_use_key_ref:
2251                      \__enumext_fake_item:
2252                      \bool_if:cT { l__enumext_show_length_#1_bool }
2253                        {
2254                          \msg_term:nnne { enumext } { list-lengths } {#1} { \int_use:N \l__enumext_level_i
2255                        }
2256                    }
2257              }
2258          }
2259      \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.

```
2260  \cs_set_protected:Npn \__enumext_tmp:n #1
2261      {
2262        \cs_new_protected:cpn { __enumext_list_arg_two_#1: }
2263          {
2264            \__enumext_calc_hspace:ccccccc
2265              { l__enumext_labelwidth_#1_dim } { l__enumext_labelsep_#1_dim }
2266              { l__enumext_listoffset_#1_dim } { l__enumext_leftmargin_tmp_#1_dim }
2267              { l__enumext_leftmargin_#1_dim } { l__enumext_itemindent_#1_dim }
2268              { l__enumext_leftmargin_tmp_#1_bool }
2269            \clist_map_inline:nn
2270              { labelsep, labelwidth, itemindent, leftmargin, rightmargin, listparindent }
2271              { \dim_set_eq:cc {####1} { l__enumext_####1_#1_dim } }
2272            \clist_map_inline:nn { topsep, parsep, partopsep, itemsep }
2273              { \skip_set_eq:cc {####1} { l__enumext_####1_#1_skip } }
2274            \skip_set_eq:Nc \parsep  { l__enumext_itemsep_#1_skip }
2275            \skip_zero:N \partopsep
2276            \usecounter { enumX#1 }
2277            \bool_lazy_and:nnTF
2278              { \str_if_eq_p:nn {#1} { vii } } { \bool_if_p:N \l__enumext_resume_vii_bool }
2279              { \setcounter { enumXvii } { \int_eval:n { \g__enumext_resume_vii_int } } }
2280              {
2281                \setcounter { enumX#1 }
2282                  { \int_eval:n { \int_use:c { l__enumext_start_#1_int } - 1 } }
2283              }
2284            \__enumext_use_key_ref_h:
2285            \str_if_eq:nnTF {#1} { vii }
2286              {
2287                \__enumext_fake_item_vii:
2288                \bool_if:cT { l__enumext_show_length_vii_bool }
2289                  { \msg_term:nnnn { enumext } { list-lengths-not-nested } { vii } { enumext* } }
2290              }
2291              {
2292                \__enumext_fake_item_viii:
2293                \bool_if:cT { l__enumext_show_length_#1_bool }
2294                  { \msg_term:nnnn { enumext } { list-lengths-not-nested } { #1 } { keyans* } }
2295              }
2296          }
2297      }
2298  \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.

```
2299 \NewDocumentEnvironment{enumext}{ O{} }
2300   {
2301     \__enumext_current_env:
2302     \__enumext_safe_exec:
2303     \__enumext_parse_keys:n {#1}
2304     \__enumext_before_list:
2305     \__enumext_start_store_level:
2306     \__enumext_start_list:nn
2307       { \tl_use:c { l__enumext_label_ \__enumext_level: _tl } }
2308       {
2309         \use:c { __enumext_list_arg_two_ \__enumext_level: : }
2310         \__enumext_before_keys_exec:
2311       }
2312     \__enumext_after_args_exec:
2313   }
2314   {
2315     \__enumext_stop_list:
2316     \__enumext_stop_store_level:
2317     \__enumext_after_list:
2318   }
```

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

__enumext_safe_exec: First check the maximum nesting level for the enumext environment and set the state of the booleans vars \l__enumext_standar_bool and \g__enumext_standar_bool to *"true"*, the latter only if the environment is NOT nested in the enumext* environment.

```
2319 \cs_new_protected:Nn \__enumext_safe_exec:
2320   {
2321     \int_incr:N \l__enumext_level_int
2322     \int_compare:nNnT { \l__enumext_level_int } > { 4 }
2323       { \msg_fatal:nn { enumext } { list-too-deep } }
2324     \bool_set_true:N \l__enumext_standar_bool
2325   }
```

(*End of definition for* __enumext_safe_exec:.)

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

```
2326 \cs_new_protected:Npn \__enumext_parse_keys:n #1
2327   {
2328     \exp_args:Ne \keys_set:nn
2329       { enumext / level-\int_use:N \l__enumext_level_int } {#1}
2330     \bool_if:NT \l__enumext_store_active_bool
2331       {
2332         \__enumext_parse_store_keys:n {#1}
2333       }
2334   }
```

(*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:.

```
2335 \cs_new_protected:Npn \__enumext_parse_store_keys:n #1
2336   {
2337     \bool_if:cF { l__enumext_store_columns_ \__enumext_level: _bool }
2338       {
2339         \regex_match:nnT { \b columns\b } {#1}
2340           {
2341             \int_set_eq:cc
2342               { l__enumext_store_columns_ \__enumext_level: _int }
2343               { l__enumext_columns_  \__enumext_level: _int }
2344             \tl_put_right:ce { l__enumext_store_opt_ \__enumext_level: _tl }
2345               {
2346                 columns = \exp_not:v { l__enumext_store_columns_ \__enumext_level: _int },
2347               }
```

```
2348                    }
2349              }
2350        \bool_if:cF { l__enumext_store_columns_sep_ \__enumext_level: _bool }
2351          {
2352            \regex_match:nnT { \b columns-sep \b} {#1}
2353              {
2354                \dim_set_eq:cc
2355                  { l__enumext_store_columns_sep_ \__enumext_level: _dim }
2356                  { l__enumext_columns_sep_  \__enumext_level: _dim }
2357                \tl_put_right:ce { l__enumext_store_opt_ \__enumext_level: _tl }
2358                  {
2359                    columns-sep = \exp_not:v { l__enumext_store_columns_sep_ \__enumext_level: _dim }
2360                  }
2361              }
2362          }
2363      }
```

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

__enumext_start_store_level: The __enumext_start_store_level: and __enumext_stop_store_level: functions activate the
__enumext_stop_store_level: level saving mechanism for storage in ⟨*sequence*⟩ of the \anskey command.
                                If enumext are nested in enumext* add __enumext_store_level_open: to preserve the stored struc-
                                ture.

```
2364  \cs_new_protected:Nn \__enumext_start_store_level:
2365    {
2366      %\bool_lazy_and:nnT
2367      \bool_lazy_all:nT
2368        {
2369          { \bool_if_p:N \l__enumext_store_active_bool }
2370          { \bool_not_p:n { \l__enumext_keyans_env_bool } }
2371          { \bool_not_p:n { \g__enumext_starred_bool } }
2372        }
2373        {
2374          \int_compare:nNnT { \l__enumext_level_int } > { 1 }
2375            {
2376              \bool_set_true:c { l__enumext_store_upper_level_ \__enumext_level: _bool }
2377              \__enumext_store_level_open:
2378            }
2379        }
2380      \bool_lazy_all:nT
2381        {
2382          { \bool_if_p:N \l__enumext_store_active_bool }
2383          { \bool_not_p:n { \l__enumext_keyans_env_bool } }
2384          { \bool_if_p:N \g__enumext_starred_bool }
2385        }
2386        {
2387          \int_compare:nNnT { \l__enumext_level_int } > { 0 }
2388            {
2389              \bool_set_true:c { l__enumext_store_upper_level_ \__enumext_level: _bool }
2390              \__enumext_store_level_open:
2391            }
2392        }
2393    }
2394  \cs_new_protected:Nn \__enumext_stop_store_level:
2395    {
2396      \bool_if:cT { l__enumext_store_upper_level_ \__enumext_level: _bool }
2397        {
2398          \__enumext_store_level_close:
2399        }
2400    }
```

(*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.

```
2401  \cs_new_protected:Nn \__enumext_before_list:
2402    {
2403      \__enumext_vspace_above:
2404      \__enumext_before_args_exec:
```

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

```
2405        \__enumext_check_ans_count:
```

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.

```
2406        \dim_compare:nNnT
2407          { \dim_use:c { l__enumext_minipage_right_ \__enumext_level: _dim } } > { \c_zero_dim }
2408          {
2409            \dim_set:cn { l__enumext_minipage_left_ \__enumext_level: _dim }
2410              {
2411                \linewidth
2412                - \dim_use:c { l__enumext_minipage_right_ \__enumext_level: _dim }
2413                - \dim_use:c { l__enumext_minipage_hsep_ \__enumext_level: _dim }
2414              }
```

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*.

```
2415            \bool_set_true:c { l__enumext_minipage_active_ \__enumext_level: _bool }
2416            \int_gincr:N \g__enumext_minipage_stat_int
2417            \__enumext_mini_addvspace:
2418            \nointerlineskip\noindent
2419            \begin{__enumext_mini_env*}
2420              { \dim_use:c { l__enumext_minipage_left_ \__enumext_level: _dim } }
2421          }
2422        \__enumext_multicols_start:
2423      }
```

(*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.

```
2424  \cs_new_protected:Nn \__enumext_multicols_start:
2425    {
2426      \int_compare:nNnT
2427        { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } > { 1 }
2428        {
2429          \dim_compare:nNnT
2430            { \dim_use:c { l__enumext_columns_sep_ \__enumext_level: _dim } } = { \c_zero_dim }
2431            {
2432              \dim_set:cn { l__enumext_columns_sep_ \__enumext_level: _dim }
2433                {
2434                  ( \dim_use:c { l__enumext_labelwidth_ \__enumext_level: _dim }
2435                    + \dim_use:c { l__enumext_labelsep_ \__enumext_level: _dim }
2436                  ) / \int_use:c { l__enumext_columns_ \__enumext_level: _int }
2437                  - \dim_use:c { l__enumext_listoffset_ \__enumext_level: _dim }
2438                }
2439            }
2440          \dim_set_eq:Nc \columnsep { l__enumext_columns_sep_ \__enumext_level: _dim  }
2441          \skip_zero:N \multicolsep
2442          \int_compare:nNnT { \l__enumext_level_int } > { 1 }
2443            {
2444              \dim_zero:N \columnseprule
2445            }
```

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.

```
2446          \bool_if:cF { l__enumext_minipage_active_ \__enumext_level: _bool }
2447            {
```

```
2448                    \__enumext_multi_addvspace:
2449                }
2450            \raggedcolumns
2451            \begin{multicols}{ \int_use:c { l__enumext_columns_ \__enumext_level: _int } }
2452        }
2453    }
```

(*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.

```
2454 \cs_new_protected:Nn \__enumext_multicols_stop:
2455    {
2456        \int_compare:nNnT
2457            { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } > { 1 }
2458            {
2459                \end{multicols}
2460                \bool_if:cF { l__enumext_minipage_active_ \__enumext_level: _bool }
2461                    {
2462                        \par\addvspace{ \skip_use:c { l__enumext_multicols_below_ \__enumext_level: _skip } }
2463                    }
2464            }
```

If the check-ans key is active, we set the boolean variable \g__enumext_check_ans_show_bool to true and copy the stored name to the variable \g__enumext_store_name_tl. These variables will be used by the function __enumext_after_env:n to display the result of the internal check answer mechanism in the terminal.

```
2465        \bool_lazy_and:nnT
2466            { \bool_if_p:N \l__enumext_check_ans_bool }
2467            { \bool_not_p:n { \g__enumext_starred_bool } }
2468            {
2469                \bool_gset_true:N \g__enumext_check_ans_show_bool
2470                \tl_gset:NV \g__enumext_store_name_tl \l__enumext_store_name_tl
2471            }
2472    }
```

(*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.

```
2473 \cs_new_protected:Nn \__enumext_after_list:
2474    {
2475        \bool_if:cTF { l__enumext_minipage_active_ \__enumext_level: _bool }
2476            {
2477                \int_compare:nNnT { \g__enumext_minipage_stat_int } = { 1 }
2478                    {
2479                        \msg_warning:nn { enumext } { missing-miniright }
2480                        \miniright
2481                    }
2482                \int_gzero:N \g__enumext_minipage_stat_int
2483                \end{__enumext_mini_env*}
2484                \par\addvspace { \l__enumext_minipage_after_skip }
2485            }
2486            { \__enumext_multicols_stop: }
```

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

```
2487        \__enumext_after_stop_list:
2488        \__enumext_vspace_below:
```

Finally save the *current value* of the counter in \g__enumext_resume_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.

```
2489        \bool_set_false:N \l__enumext_standar_bool
2490        %\bool_gset_false:N \g__enumext_standar_bool
2491        \int_gset_eq:NN \g__enumext_resume_int \value{enumXi}
2492        \int_if_exist:cT { g__enumext_resume_ \l__enumext_store_name_tl _int }
```

```
2493        {
2494            \int_gset_eq:cN
2495                { g__enumext_resume_ \l__enumext_store_name_tl _int }
2496                { \value{enumXi} }
2497        }
2498    }
```

(*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.

```
2499 \__enumext_after_env:nn {enumext}
2500    {
2501        \int_compare:nNnT { \l__enumext_level_int } = { 0 }
2502            {
2503                \bool_if:NT \g__enumext_check_ans_show_bool
2504                    {
2505                        \__enumext_check_ans_active:
2506                    }
2507                \bool_gset_false:N \g__enumext_standar_bool
2508                \bool_gset_false:N \g__enumext_check_ans_show_bool
2509                \tl_gclear:N \g__enumext_store_name_tl
2510            }
2511    }
```

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.

```
2512 \NewDocumentEnvironment{keyans}{ O{} }
2513    {
2514        \__enumext_keyans_safe_exec:
2515        \__enumext_keyans_parse_keys:n {#1}
2516        \__enumext_before_list_v:
2517        \__enumext_start_list:nn
2518            { \tl_use:N \l__enumext_label_v_tl }
2519            {
2520                \__enumext_list_arg_two_v:
2521                \__enumext_before_keys_exec_v:
2522            }
2523        \__enumext_after_args_exec_v:
2524    }
2525    {
2526        \__enumext_keyans_check_ans:nn { item }{ keyans }
2527        \__enumext_stop_list:
2528        \__enumext_after_list_v:
2529    }
```

(*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.

```
2530 \cs_new_protected:Nn \__enumext_keyans_safe_exec:
2531    {
2532        \bool_if:NF \l__enumext_store_active_bool
2533            {
2534                \msg_error:nnnn { enumext } { wrong-place }{ keyans }{ save-ans }
2535            }
2536        \int_incr:N \l__enumext_keyans_level_int
2537        \bool_set_true:N \l__enumext_keyans_env_bool
2538        % Set false for interfering with enumext nested in keyans (yes, its possible and crayze)
2539        \bool_set_false:N \l__enumext_store_active_bool
2540        \int_compare:nNnT { \l__enumext_keyans_level_int } > { 1 }
2541            {
2542                \msg_error:nn { enumext } { keyans-nested }
2543            }
2544        \int_compare:nNnT { \l__enumext_level_int } > { 1 }
2545            {
```

```
2546              \msg_error:nn { enumext } { keyans-wrong-level }
2547          }
2548      }
```

(*End of definition for __enumext_keyans_safe_exec:.*)

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

```
2549 \cs_new_protected:Npn \__enumext_keyans_parse_keys:n #1
2550    {
2551      \keys_set:nn { enumext / keyans } {#1}
2552    }
```

(*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.

```
2553 \cs_new_protected:Nn \__enumext_before_list_v:
2554    {
2555      \__enumext_vspace_above_v:
2556      \__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.

```
2557      \dim_compare:nNnT { \l__enumext_minipage_right_v_dim } > { \c_zero_dim }
2558        {
2559          \dim_set:Nn \l__enumext_minipage_left_v_dim
2560            {
2561              \linewidth - \l__enumext_minipage_right_v_dim - \l__enumext_minipage_hsep_v_dim
2562            }
```

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*.

```
2563          \bool_set_true:N \l__enumext_minipage_active_v_bool
2564          \int_gincr:N \g__enumext_minipage_stat_int
2565          \__enumext_keyans_mini_addvspace:
2566          \nointerlineskip\noindent
2567          \begin{__enumext_mini_env*}{ \l__enumext_minipage_left_v_dim }
2568        }
```

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

```
2569      \__enumext_keyans_multicols_start:
2570    }
```

(*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.

```
2571 \cs_new_protected:Nn \__enumext_keyans_multicols_start:
2572    {
2573      \int_compare:nNnT { \l__enumext_columns_v_int } > { 1 }
2574        {
```

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

```
2575          \dim_compare:nNnT { \l__enumext_columns_sep_v_dim } = { \c_zero_dim }
2576            {
2577              \dim_set:Nn \l__enumext_columns_sep_v_dim
2578                {
2579                  (
2580                    \l__enumext_labelwidth_v_dim + \l__enumext_labelsep_v_dim
2581                  ) / \l__enumext_columns_v_int
2582                  - \l__enumext_listoffset_v_dim
2583                }
2584            }
2585          \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).

```
2586          \skip_zero:N \multicolsep
2587          \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.

```
2588          \bool_if:NF \l__enumext_minipage_active_v_bool
2589            {
2590              \__enumext_keyans_multi_addvspace:
2591            }
2592          \raggedcolumns
2593          \begin{multicols}{ \l__enumext_columns_v_int }
2594        }
2595    }
```

(*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.

```
2596  \cs_new_protected:Nn \__enumext_keyans_multicols_stop:
2597    {
2598      \int_compare:nNnT { \l__enumext_columns_v_int } > { 1 }
2599        {
2600          \end{multicols}
2601          \bool_if:NF \l__enumext_minipage_active_v_bool
2602            {
2603              \par\addvspace{ \l__enumext_multicols_below_v_skip }
2604            }
2605        }
2606    }
```

(*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.

```
2607  \cs_new_protected:Nn \__enumext_after_list_v:
2608    {
2609      \bool_if:NTF \l__enumext_minipage_active_v_bool
2610        {
2611          \int_compare:nNnT { \g__enumext_minipage_stat_int } = { 1 }
2612            {
2613              \msg_warning:nn { enumext } { missing-miniright }
2614              \miniright
2615            }
2616          \int_gzero:N \g__enumext_minipage_stat_int
2617          \end{__enumext_mini_env*}
2618          \par\addvspace{ \l__enumext_minipage_after_skip }
2619        }
2620        { \__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.

```
2621      \bool_set_false:N \l__enumext_keyans_env_bool
2622      \__enumext_after_stop_list_v:
2623      \__enumext_vspace_below_v:
2624    }
```

(*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.
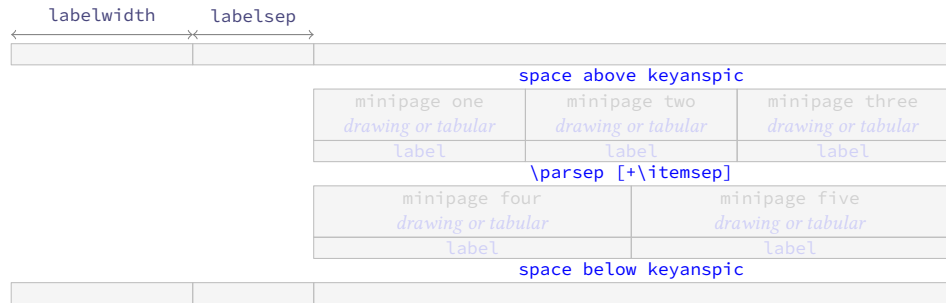
Figure 12: Representation of the keyanspic spacing in enumext.

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.

```
2625 \NewDocumentCommand \anspic { s o +m }
2626     {
```

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

```
2627        \bool_if:NF \l__enumext_store_active_bool
2628          {
2629            \msg_error:nnnn { enumext } { wrong-place }{ keyanspic }{ save-ans }
2630          }
2631        \int_compare:nNnT { \l__enumext_level_int } > { 1 }
2632          {
2633            \msg_error:nn { enumext } { keyanspic-wrong-level }
2634          }
2635        \int_compare:nNnT { \l__enumext_keyans_level_int } = { 1 }
2636          {
2637            \msg_error:nnnn { enumext } { command-wrong-place }{ anspic }{ keyans }
2638          }
```

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.

```
2639        \seq_put_right:Nn \l__enumext_keyans_pic_body_seq
2640          {
2641            \__enumext_keyans_anspic_code:nnn { #1 } { #2 } { #3 }
2642          }
2643     }
```

(*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.

```
2644 \cs_new_protected:Nn \__enumext_keyans_anspic_code:nnn
2645     {
2646        \stepcounter { enumXvi }
2647        #3 \\
2648        \bool_if:nT { #1 }
2649          {
2650            \__enumext_keyans_addto_prop:n { #2 }
2651            \__enumext_keyans_store_ref:
2652            \__enumext_keyans_addto_seq:n { #2 }
2653            \bool_lazy_or:nnT
2654              { \bool_if_p:N \l__enumext_show_answer_bool }
2655              { \bool_if_p:N \l__enumext_show_position_bool }
```

```
2656              {
2657                  \tl_set_eq:NN \l__enumext_label_v_tl \l__enumext_label_vi_tl
2658                  \__enumext_keyans_show_left:n { #2 }
2659                  \tl_set_eq:NN \l__enumext_label_vi_tl \l__enumext_label_v_tl
2660              }
2661          }
2662      \tl_use:N \l__enumext_label_font_style_v_tl
2663      \__enumext_wrapper_label_v:n { \l__enumext_label_vi_tl }
2664  }
```

(*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.

```
2665  \NewDocumentEnvironment{keyanspic}{ o }
2666    {
2667      \__enumext_keyans_pic_safe_exec:
2668      \__enumext_start_list:nn
2669        { }
2670        {
2671            \__enumext_keyans_pic_arg_two:
2672        }
```

We apply the "adjusted" vertical spacing above the environment

```
2673          \vspace { \l__enumext_keyans_pic_above_skip }
2674  }
```

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.

```
2675    {
2676      \tl_if_novalue:nTF { #1 }
2677        {
2678            \__enumext_keyans_pic_do:e { \seq_count:N \l__enumext_keyans_pic_body_seq }
2679        }
2680        { \__enumext_keyans_pic_do:n { #1 } }
2681      \__enumext_stop_list:
2682      \__enumext_keyans_check_ans:nn { anspic } { keyanspic }
2683      \setcounter { enumXvi } { 0 }
2684      \vspace { \l__enumext_topsep_v_skip }
2685      %\bool_set_false:N \l__enumext_store_active_bool
2686  }
```

(*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.

```
2687  \cs_new_protected:Nn \__enumext_keyans_pic_safe_exec:
2688    {
2689      \int_incr:N \l__enumext_keyans_pic_level_int
2690      \int_compare:nNnT { \l__enumext_keyans_pic_level_int } > { 1 }
2691        {
2692            \msg_error:nn { enumext } { keyanspic-nested }
2693        }
2694  }
```

(*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.

```
2695  \cs_new_protected:Npn \__enumext_keyans_pic_skip_abs:N #1
2696    {
2697      \dim_compare:nNnT { #1 } < { 0pt }
2698        { \skip_set:Nn #1 { -#1 } }
2699  }
```

(*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.

```
2700  \cs_new_protected:Nn \__enumext_keyans_pic_arg_two:
2701    {
```

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.

```
2702      \bool_set_false:N \l__enumext_leftmargin_tmp_v_bool
2703      \__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.

```
2704      \skip_add:Nn \parsep { \itemsep }
2705      \dim_add:Nn  \leftmargin { -\labelwidth - \labelsep }
2706      \dim_zero:N  \labelwidth
2707      \dim_zero:N  \listparindent
2708      \dim_zero:N  \labelsep
2709      \skip_zero:N \partopsep
2710      \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.

```
2711      \__enumext_keyans_pic_skip_abs:N \parsep
2712      \skip_set:Nn \l__enumext_keyans_pic_above_skip
2713        {
2714          \box_dp:N \strutbox
2715          + \l__enumext_topsep_v_skip
2716          - \parsep
2717        }
2718      \__enumext_item_std:w \scan_stop:
2719    }
```

(*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.

```
2720  \cs_new_protected:Nn \__enumext_keyans_pic_do:n
2721    {
2722      \clist_map_function:nN { #1 } \__enumext_keyans_pic_row:n
2723    }
2724  \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.

```
2725  \cs_new_protected:Nn \__enumext_keyans_pic_row:n
2726    {
2727      \dim_set:Nn \l__enumext_keyans_pic_width_dim { \linewidth / #1 }
2728      \int_set:Nn \l__enumext_keyans_pic_above_int { \l__enumext_keyans_pic_below_int }
2729      \int_set:Nn \l__enumext_keyans_pic_below_int { \l__enumext_keyans_pic_above_int + #1 }
2730      \int_step_inline:nnn
2731        { \l__enumext_keyans_pic_above_int + 1 }
2732        { \l__enumext_keyans_pic_below_int }
2733        {
2734          \__enumext_minipage:w [ b ]{ \l__enumext_keyans_pic_width_dim }
2735            \centering
2736            \seq_item:Nn \l__enumext_keyans_pic_body_seq { ##1 }
2737          \__enumext_endminipage:
2738        }
2739      \par
2740    }
```

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

10.35 The `enumext*` environment

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.

```
2741 \cs_new_protected:Nn \__enumext_starred_columns_set_vii:
2742   {
2743     \dim_compare:nNnT { \l__enumext_columns_sep_vii_dim } = { \c_zero_dim }
2744       {
2745         \dim_set:Nn \l__enumext_columns_sep_vii_dim
2746           {
2747             ( \l__enumext_labelwidth_vii_dim + \l__enumext_labelsep_vii_dim )
2748             / \l__enumext_columns_vii_int
2749           }
2750       }
2751     \int_set:Nn \l__enumext_tmpa_vii_int { \l__enumext_columns_vii_int - \c_one_int }
2752     \dim_set:Nn \l__enumext_item_width_vii_dim
2753       {
2754         ( \linewidth - \l__enumext_columns_sep_vii_dim * \l__enumext_tmpa_vii_int )
2755         / \l__enumext_columns_vii_int - \l__enumext_labelwidth_vii_dim
2756         - \l__enumext_labelsep_vii_dim
2757       }
2758     \dim_zero_new:N \itemwidth
2759   }
```

(*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`.

```
2760 \cs_new_protected:Npn \__enumext_starred_joined_item_vii:n #1
2761   {
2762     \int_set:Nn \l__enumext_joined_item_vii_int {#1}
2763     \int_compare:nNnT { \l__enumext_joined_item_vii_int } > { \l__enumext_columns_vii_int }
2764       {
2765         \msg_warning:nnee { enumext } { item-joined }
2766           { \int_use:N \l__enumext_joined_item_vii_int }
2767           { \int_use:N \l__enumext_columns_vii_int }
2768         \int_set:Nn \l__enumext_joined_item_vii_int
2769           {
2770             \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + \c_one_int
2771           }
2772       }
2773     \int_compare:nNnT
2774       { \l__enumext_joined_item_vii_int }
2775         >
2776       { \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + \c_one_int }
2777       {
2778         \msg_warning:nnee { enumext } { item-joined-columns }
2779           { \int_use:N \l__enumext_joined_item_vii_int }
2780           {
2781             \int_eval:n
2782               { \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + \c_one_int }
2783           }
2784         \int_set:Nn \l__enumext_joined_item_vii_int
2785           {
2786             \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + \c_one_int
2787           }
2788       }
```

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

```
2789      \int_compare:nNnTF { \l__enumext_joined_item_vii_int } > { \c_one_int }
2790        {
2791          \int_set_eq:NN \l__enumext_joined_item_aux_vii_int \l__enumext_joined_item_vii_int
2792          \int_decr:N \l__enumext_joined_item_aux_vii_int
2793          \int_add:Nn \l__enumext_item_column_pos_vii_int { \l__enumext_joined_item_aux_vii_int }
2794          \int_gadd:Nn \g__enumext_item_count_all_vii_int { \l__enumext_joined_item_aux_vii_int }
2795          \dim_set:Nn \l__enumext_joined_width_vii_dim
2796            {
2797              \l__enumext_item_width_vii_dim * \l__enumext_joined_item_vii_int
2798              + (  \l__enumext_labelwidth_vii_dim + \l__enumext_labelsep_vii_dim
2799                + \l__enumext_columns_sep_vii_dim
2800              )*\l__enumext_joined_item_aux_vii_int
2801            }
2802          \dim_set_eq:NN \itemwidth \l__enumext_joined_width_vii_dim
2803        }
2804        {
2805          \dim_set_eq:NN \l__enumext_joined_width_vii_dim \l__enumext_item_width_vii_dim
2806          \dim_set_eq:NN \itemwidth \l__enumext_item_width_vii_dim
2807        }
2808    }
```

(*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.

```
2809  \cs_new_protected:Nn \__enumext_start_mini_vii:
2810    {
2811      \dim_compare:nNnT { \l__enumext_minipage_right_vii_dim } > { \c_zero_dim }
2812        {
2813          \dim_set:Nn \l__enumext_minipage_left_vii_dim
2814            {
2815              \linewidth
2816              - \l__enumext_minipage_right_vii_dim
2817              - \l__enumext_minipage_hsep_vii_dim
2818            }
2819          \bool_set_true:N \l__enumext_minipage_active_vii_bool
2820          \dim_gset_eq:NN
2821            \g__enumext_minipage_right_vii_dim
2822            \l__enumext_minipage_right_vii_dim
2823          \__enumext_mini_addvspace_vii:
2824          \nointerlineskip\noindent
2825          \begin{__enumext_mini_env*}{ \l__enumext_minipage_left_vii_dim }
2826        }
2827    }
```

(*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"*.

```
2828  \cs_new_protected:Nn \__enumext_stop_mini_vii:
2829    {
2830      \bool_if:NT \l__enumext_minipage_active_vii_bool
2831        {
2832          \end{__enumext_mini_env*}
2833          \hfill
2834          \bool_gset_true:N \g__enumext_minipage_active_vii_bool
2835        }
2836    }
```

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"*.

```
2837  \__enumext_after_env:nn {enumext*}
2838    {
2839      \bool_if:NT \g__enumext_minipage_active_vii_bool
2840        {
```

```
2841          \begin{__enumext_mini_env*}{ \g__enumext_minipage_right_vii_dim }
2842            \par\addvspace { \g__enumext_minipage_right_skip }
2843            \bool_if:NF \g__enumext_minipage_center_vii_bool
2844              {
2845                \centering
2846              }
2847            \tl_use:N \g__enumext_miniright_code_vii_tl % the code
2848          \end{__enumext_mini_env*}
2849          \par\addvspace{ \g__enumext_minipage_after_skip }
2850        }
2851      \bool_gset_false:N \g__enumext_minipage_active_vii_bool
2852      \bool_gset_true:N \g__enumext_minipage_center_vii_bool
2853      \tl_gclear:N \g__enumext_miniright_code_vii_tl
2854      \dim_gzero:N \g__enumext_minipage_right_vii_dim
2855    }
```

(*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.

```
2856  \NewDocumentEnvironment{enumext*}{ o }
2857    {
2858      \__enumext_current_env:
2859      \__enumext_safe_exec_vii:
2860      \__enumext_parse_keys_vii:n {#1}
2861      \__enumext_before_list_vii:
2862      \__enumext_start_store_level_vii:
2863      \__enumext_start_list:nn { }
2864        {
2865          \__enumext_list_arg_two_vii:
2866          \__enumext_before_keys_exec_vii:
2867        }
2868      \__enumext_starred_columns_set_vii:
2869      \item[] \scan_stop:
2870      \cs_set_eq:NN \__enumext_stop_item_tmp_vii: \noindent
2871      \cs_set_eq:NN \item \__enumext_start_item_tmp_vii:
2872    }
2873    {
2874      \__enumext_stop_item_tmp_vii:
2875      \__enumext_remove_extra_parsep_vii:
2876      \__enumext_stop_list:
2877      \__enumext_stop_store_level_vii:
2878      \__enumext_after_list_vii:
2879    }
```

(*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.

```
2880  \cs_new_protected:Nn \__enumext_safe_exec_vii:
2881    {
2882      \int_incr:N \l__enumext_level_h_int
2883      \int_compare:nNnT { \l__enumext_level_h_int } > { 1 }
2884        {
2885          \msg_error:nn { enumext } { nested }
2886        }
2887      \bool_set_true:N \l__enumext_starred_bool
2888    }
```

(*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.

```
2889  \cs_new_protected:Npn \__enumext_parse_keys_vii:n #1
2890    {
2891      \tl_if_novalue:nF {#1}
2892        {
2893          \keys_set:nn { enumext / enumext* } {#1}
2894          \bool_if:NT \l__enumext_store_active_bool
```

```
2895              {
2896                  \__enumext_parse_store_keys_vii:n {#1}
2897              }
2898          }
2899      }
```

(*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:.

```
2900  \cs_new_protected:Npn \__enumext_parse_store_keys_vii:n #1
2901      {
2902          \bool_if:NF \l__enumext_store_columns_vii_bool
2903              {
2904                  \regex_match:nnT { \b columns\b } {#1}
2905                      {
2906                          \int_set_eq:NN
2907                              \l__enumext_store_columns_vii_int
2908                              \l__enumext_columns_vii_int
2909                          \tl_put_right:Ne \l__enumext_store_opt_vii_tl
2910                              {
2911                                  columns = \exp_not:V \l__enumext_store_columns_vii_int ,
2912                              }
2913                      }
2914              }
2915          \bool_if:NF \l__enumext_store_columns_sep_vii_bool
2916              {
2917                  \regex_match:nnT { \b columns-sep \b} {#1}
2918                      {
2919                          \dim_set_eq:NN
2920                              \l__enumext_store_columns_sep_vii_dim
2921                              \l__enumext_columns_sep_vii_dim
2922                          \tl_put_right:Ne \l__enumext_store_opt_vii_tl
2923                              {
2924                                  columns-sep = \exp_not:V \l__enumext_store_columns_sep_vii_dim,
2925                              }
2926                      }
2927              }
2928      }
```

(*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.

```
2929  \cs_new_protected:Nn \__enumext_before_list_vii:
2930      {
2931          \__enumext_vspace_above_vii:
2932          \__enumext_check_ans_count: % need by chek-ans
2933          \__enumext_before_args_exec_vii:
2934          \__enumext_start_mini_vii:
2935      }
```

(*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.

```
2936  \cs_new_protected:Nn \__enumext_after_list_vii:
2937      {
2938          \__enumext_stop_mini_vii:
2939          \__enumext_after_stop_list_vii:
2940          \__enumext_vspace_below_vii:
2941          \int_gset_eq:NN \g__enumext_resume_vii_int \value{enumXvii}
```

```
2942    \int_if_exist:cT { g__enumext_resume_ \l__enumext_store_name_tl _int }
2943      {
2944        \int_gset_eq:cN
2945          { g__enumext_resume_ \l__enumext_store_name_tl _int }
2946          { \value{enumXvii} }
2947      }
2948    \bool_lazy_and:nnT
2949      { \bool_if_p:N \g__enumext_starred_bool }
2950      { \bool_if_p:N \l__enumext_check_ans_bool }
2951      {
2952        \bool_gset_true:N \g__enumext_check_ans_show_h_bool
2953        \tl_gset:NV \g__enumext_store_name_tl \l__enumext_store_name_tl
2954      }
2955    %\bool_gset_false:N \g__enumext_starred_bool
2956    \bool_set_false:N \l__enumext_starred_bool
2957  }
```

(*End of definition for* __enumext_after_list_vii:.)

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

```
2958  \cs_new_protected:Nn \__enumext_start_store_level_vii:
2959    {
2960      \bool_if:NT \l__enumext_store_active_bool
2961        {
2962          \int_compare:nNnT { \l__enumext_level_int } > { \c_zero_int }
2963            {
2964              \__enumext_store_level_open_vii:
2965            }
2966        }
2967    }
2968  \cs_new_protected:Nn \__enumext_stop_store_level_vii:
2969    {
2970      \bool_if:NT \l__enumext_store_active_bool
2971        {
2972          \int_compare:nNnT { \l__enumext_level_int } > { \c_zero_int }
2973            {
2974              \__enumext_store_level_close_vii:
2975            }
2976        }
2977    }
```

(*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.

```
2978  \cs_new_protected_nopar:Nn \__enumext_start_item_tmp_vii:
2979    {
2980      \__enumext_stop_item_tmp_vii:
2981      \int_incr:N \l__enumext_item_column_pos_vii_int
2982      \int_gincr:N \g__enumext_item_count_all_vii_int
2983      \__enumext_item_peek_args_vii:
2984    }
```

(*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).

```
2985  \cs_new_protected:Nn \__enumext_item_peek_args_vii:
2986    {
2987      \peek_meaning:NTF (
2988        { \__enumext_joined_item_vii:w }
2989        { \__enumext_joined_item_vii:w (1) }
2990    }
```

(*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.

```
2991 \cs_new_protected:Npn \__enumext_joined_item_vii:w (#1)
2992   {
2993     \__enumext_starred_joined_item_vii:n {#1}
2994     \peek_meaning_remove:NTF *
2995       { \__enumext_starred_item_vii:w  }
2996       { \__enumext_standard_item_vii:w }
2997   }
```

(*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].

```
2998 \cs_new_protected:Npn \__enumext_standard_item_vii:w
2999   {
3000     \bool_set_false:N \l__enumext_item_starred_vii_bool
3001     \peek_meaning:NTF [
3002       {
3003         \bool_set_eq:NN
3004           \l__enumext_wrap_label_vii_bool
3005           \l__enumext_wrap_label_opt_vii_bool
3006         \__enumext_start_item_vii:w
3007       }
3008       {
3009         \bool_set_true:N \l__enumext_wrap_label_vii_bool
3010         \legacy_if_set_true:n { @noitemarg }
3011         \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
3012       }
3013   }
```

(*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*⟩].

```
3014 \cs_new_protected:Npn \__enumext_starred_item_vii:w
3015   {
3016     \bool_set_true:N \l__enumext_item_starred_vii_bool
3017     \bool_set_true:N \l__enumext_wrap_label_vii_bool
3018     \peek_meaning:NTF [
3019       { \__enumext_starred_item_vii_aux_i:w }
3020       { \__enumext_starred_item_vii_aux_ii:w }
3021   }
3022 \cs_new_protected:Npn \__enumext_starred_item_vii_aux_i:w [#1]
3023   {
3024     \tl_gset:Nn \g__enumext_item_symbol_aux_vii_tl {#1}
3025     \__enumext_starred_item_vii_aux_ii:w
3026   }
3027 \cs_new_protected:Npn \__enumext_starred_item_vii_aux_ii:w
3028   {
3029     \peek_meaning:NTF [
3030       { \__enumext_starred_item_vii_aux_iii:w }
3031       {
3032         \dim_set_eq:NN
3033           \l__enumext_item_symbol_sep_vii_dim
3034           \l__enumext_labelsep_vii_dim
3035         \legacy_if_set_true:n { @noitemarg }
3036         \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
3037       }
3038   }
3039 \cs_new_protected:Npn \__enumext_starred_item_vii_aux_iii:w [#1]
```

```
3040    {
3041      \dim_set:Nn \l__enumext_item_symbol_sep_vii_dim {#1}
3042      \legacy_if_set_true:n { @noitemarg }
3043      \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
3044    }
```

(*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.

```
3045  \cs_new_protected_nopar:Npn \__enumext_start_item_vii:w [#1]
3046    {
3047      \cs_set_eq:NN \__enumext_stop_item_tmp_vii: \__enumext_stop_item_vii:
3048      \legacy_if:nT { @noitemarg }
3049        {
3050          \legacy_if_set_false:n { @noitemarg }
3051          \legacy_if:nT { @nmbrlist }
3052            {
3053              \bool_if:NT \l__enumext_hyperref_bool
3054                {
3055                  \legacy_if_set_true:n { @hyper@item }
3056                }
3057              \refstepcounter{enumXvii}
3058              \bool_if:NT \l__enumext_check_ans_bool
3059                {
3060                  \int_gincr:N \g__enumext_count_item_number_int
3061                }
3062            }
3063        }
```

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.

```
3064        \group_begin:
3065          \lrbox{ \l__enumext_item_text_vii_box }
3066            \bool_if:NF \l__enumext_footnotes_key_bool
3067              {
3068                \__enumext_renew_footnote:
3069              }
3070            \bool_if:NT \l__enumext_item_starred_vii_bool
3071              {
3072                \tl_if_blank:VT \g__enumext_item_symbol_aux_vii_tl
3073                  {
3074                    \tl_gset_eq:NN
3075                      \g__enumext_item_symbol_aux_vii_tl \l__enumext_item_symbol_vii_tl
3076                  }
3077                \mode_leave_vertical:
3078                \skip_horizontal:n { -\l__enumext_item_symbol_sep_vii_dim }
3079                \makebox[ 0pt ][ r ]{ \g__enumext_item_symbol_aux_vii_tl }
3080                \skip_horizontal:N \l__enumext_item_symbol_sep_vii_dim
3081                \tl_gclear:N \g__enumext_item_symbol_aux_vii_tl
3082              }
3083            \group_begin:
3084              \tl_use:N \l__enumext_label_font_style_vii_tl
3085              \bool_if:NTF \l__enumext_wrap_label_vii_bool
3086                {
3087                  \makebox[ \l__enumext_labelwidth_vii_dim ][ \l__enumext_align_label_vii_str ]
3088                    { \__enumext_wrapper_label_vii:n {#1} }
3089                }
3090                {
```

```
3091                    \makebox[ \l__enumext_labelwidth_vii_dim ][ \l__enumext_align_label_vii_str ]{ #1 }
3092                }
3093            \group_end:
3094            \skip_horizontal:N \l__enumext_labelsep_vii_dim
3095            \tl_use:N \l__enumext_after_list_args_vii_tl
3096            \__enumext_minipage:w [ t ]{ \l__enumext_joined_width_vii_dim  }
3097              \skip_set_eq:NN \parindent \l__enumext_listparindent_vii_dim
3098              \skip_set_eq:NN \parskip \l__enumext_parsep_vii_skip
3099              \tl_use:N \l__enumext_fake_item_indent_vii_tl
3100        }
```

(*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.

```
3101  \cs_new_protected_nopar:Nn \__enumext_stop_item_vii:
3102    {
3103            \__enumext_endminipage:
3104          \endlrbox
3105        \group_end:
3106        \box_set_wd:Nn \l__enumext_item_text_vii_box
3107          {
3108            \l__enumext_joined_width_vii_dim
3109            + \l__enumext_labelwidth_vii_dim
3110            + \l__enumext_labelsep_vii_dim
3111          }
3112        \int_set:Nn \hbadness { 10000 }
3113        \box_use:N \l__enumext_item_text_vii_box
3114        \bool_if:NF \l__enumext_footnotes_key_bool
3115          {
3116            \__enumext_print_footnote:
3117          }
3118        \int_compare:nNnTF { \l__enumext_item_column_pos_vii_int } = { \l__enumext_columns_vii_int }
3119          {
3120            \par\noindent
3121            \int_zero:N \l__enumext_item_column_pos_vii_int
3122          }
3123          { \hspace{ \l__enumext_columns_sep_vii_dim } }
3124    }
```

(*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.

```
3125  \cs_new_protected:Nn \__enumext_remove_extra_parsep_vii:
3126    {
3127      \int_compare:nNnT
3128        {
3129          \int_mod:nn {  \g__enumext_item_count_all_vii_int } { \l__enumext_columns_vii_int }
3130        }
3131        =
3132        { \c_zero_int }
3133        {
3134          \par
3135          \vspace{ -\l__enumext_itemsep_vii_skip }
3136          \int_gzero:N \g__enumext_item_count_all_vii_int
3137        }
3138    }
```

(*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.

```
3139  \__enumext_after_env:nn {enumext*}
3140    {
3141      \int_compare:nNnT { \l__enumext_level_int } = { 0 }
3142        {
3143          \bool_if:NT \g__enumext_check_ans_show_h_bool
3144            {
3145              \__enumext_check_ans_active:
```

```
3146        }
3147      \bool_gset_false:N \g__enumext_starred_bool
3148      \bool_gset_false:N \g__enumext_check_ans_show_h_bool
3149      \tl_gclear:N \g__enumext_store_name_tl
3150    }
3151  }
```

10.36 The keyans* environment

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.

```
3152 \cs_new_protected:Nn \__enumext_starred_columns_set_viii:
3153   {
3154     \dim_compare:nNnT { \l__enumext_columns_sep_viii_dim } = { \c_zero_dim }
3155       {
3156         \dim_set:Nn \l__enumext_columns_sep_viii_dim
3157           {
3158             ( \l__enumext_labelwidth_viii_dim + \l__enumext_labelsep_viii_dim )
3159             / \l__enumext_columns_viii_int
3160           }
3161       }
3162     \int_set:Nn \l__enumext_tmpa_viii_int { \l__enumext_columns_viii_int - \c_one_int }
3163     \dim_set:Nn \l__enumext_item_width_viii_dim
3164       {
3165         ( \linewidth - \l__enumext_columns_sep_viii_dim * \l__enumext_tmpa_viii_int )
3166         / \l__enumext_columns_viii_int - \l__enumext_labelwidth_viii_dim
3167         - \l__enumext_labelsep_viii_dim
3168       }
3169     \dim_zero_new:N \itemwidth
3170   }
```

(*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.

```
3171 \cs_new_protected:Npn \__enumext_starred_joined_item_viii:n #1
3172   {
3173     \int_set:Nn \l__enumext_joined_item_viii_int {#1}
3174     \int_compare:nNnT { \l__enumext_joined_item_viii_int } > { \l__enumext_columns_viii_int }
3175       {
3176         \msg_warning:nnee { enumext } { item-joined }
3177           { \int_use:N \l__enumext_joined_item_viii_int }
3178           { \int_use:N \l__enumext_columns_viii_int }
3179         \int_set:Nn \l__enumext_joined_item_viii_int
3180           {
3181             \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + \c_one_int
3182           }
3183       }
3184     \int_compare:nNnT
3185       { \l__enumext_joined_item_viii_int }
3186       >
3187       { \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + \c_one_int }
3188       {
3189         \msg_warning:nnee { enumext } { item-joined-columns }
3190           { \int_use:N \l__enumext_joined_item_viii_int }
3191           {
3192             \int_eval:n
3193               { \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + \c_one_int }
3194           }
3195         \int_set:Nn \l__enumext_joined_item_viii_int
3196           {
3197             \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + \c_one_int
3198           }
3199       }
3200     \int_compare:nNnTF { \l__enumext_joined_item_viii_int } > { \c_one_int }
3201       {
3202         \int_set_eq:NN \l__enumext_joined_item_aux_viii_int \l__enumext_joined_item_viii_int
3203         \int_decr:N \l__enumext_joined_item_aux_viii_int
3204         \int_add:Nn \l__enumext_item_column_pos_viii_int { \l__enumext_joined_item_aux_viii_int }
3205         \int_gadd:Nn \g__enumext_item_count_all_viii_int { \l__enumext_joined_item_aux_viii_int }
```

```
3206        \dim_set:Nn \l__enumext_joined_width_viii_dim
3207          {
3208            \l__enumext_item_width_viii_dim * \l__enumext_joined_item_viii_int
3209          + (  \l__enumext_labelwidth_viii_dim + \l__enumext_labelsep_viii_dim
3210             + \l__enumext_columns_sep_viii_dim
3211            )*\l__enumext_joined_item_aux_viii_int
3212          }
3213        \dim_set_eq:NN \itemwidth \l__enumext_joined_width_viii_dim
3214      }
3215      {
3216        \dim_set_eq:NN \l__enumext_joined_width_viii_dim \l__enumext_item_width_viii_dim
3217        \dim_set_eq:NN \itemwidth \l__enumext_item_width_viii_dim
3218      }
3219    }
```

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

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

```
3220  \cs_new_protected:Nn \__enumext_start_mini_viii:
3221    {
3222      \dim_compare:nNnT { \l__enumext_minipage_right_viii_dim } > { \c_zero_dim }
3223        {
3224          \dim_set:Nn \l__enumext_minipage_left_viii_dim
3225            {
3226              \linewidth
3227              - \l__enumext_minipage_right_viii_dim
3228              - \l__enumext_minipage_hsep_viii_dim
3229            }
3230          \bool_set_true:N \l__enumext_minipage_active_viii_bool
3231          \dim_gset_eq:NN
3232            \g__enumext_minipage_right_viii_dim
3233            \l__enumext_minipage_right_viii_dim
3234          \__enumext_mini_addvspace_viii:
3235          \nointerlineskip\noindent
3236          \begin{__enumext_mini_env*}{ \l__enumext_minipage_left_viii_dim }
3237        }
3238    }
3239  \cs_new_protected:Nn \__enumext_stop_mini_viii:
3240    {
3241      \bool_if:NT \l__enumext_minipage_active_viii_bool
3242        {
3243          \end{__enumext_mini_env*}
3244          \hfill
3245          \bool_gset_true:N \g__enumext_minipage_active_viii_bool
3246        }
3247    }
3248  \__enumext_after_env:nn {keyans*}
3249    {
3250      \bool_if:NT \g__enumext_minipage_active_viii_bool
3251        {
3252          \begin{__enumext_mini_env*}{ \g__enumext_minipage_right_viii_dim }
3253            \par\addvspace { \g__enumext_minipage_right_skip }
3254            \bool_if:NF \g__enumext_minipage_center_viii_bool
3255              {
3256                \centering
3257              }
3258            \tl_use:N \g__enumext_miniright_code_viii_tl % the code
3259          \end{__enumext_mini_env*}
3260          \par\addvspace{ \g__enumext_minipage_after_skip }
3261        }
3262      \bool_gset_false:N \g__enumext_minipage_active_viii_bool
3263      \bool_gset_true:N \g__enumext_minipage_center_viii_bool
3264      \tl_gclear:N \g__enumext_miniright_code_viii_tl
3265      \dim_gzero:N \g__enumext_minipage_right_viii_dim
3266    }
```

(*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.

```
3267  \NewDocumentEnvironment{keyans*}{ o }
3268    {
3269      \__enumext_safe_exec_viii:
3270      \__enumext_parse_keys_viii:n {#1}
3271      \__enumext_before_list_viii:
3272      \__enumext_start_list:nn { }
3273        {
3274          \__enumext_list_arg_two_viii:
3275          \__enumext_before_keys_exec_viii:
3276        }
3277      \__enumext_starred_columns_set_viii:
3278      \item[] \scan_stop:
3279      \cs_set_eq:NN \__enumext_stop_item_tmp_viii: \noindent
3280      \cs_set_eq:NN \item \__enumext_start_item_tmp_viii:
3281    }
3282    {
3283      \__enumext_stop_item_tmp_viii:
3284      \__enumext_remove_extra_parsep_viii:
3285      \__enumext_keyans_check_ans:nn { item }{ keyans* }
3286      \__enumext_stop_list:
3287      \__enumext_after_list_viii:
3288    }
```

(*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.

```
3289  \cs_new_protected:Nn \__enumext_safe_exec_viii:
3290    {
3291      \int_incr:N \l__enumext_keyans_level_h_int
3292      \int_compare:nNnT { \l__enumext_keyans_level_h_int } > { 1 }
3293        {
3294          \msg_error:nn { enumext } { nested }
3295        }
3296      % Set false for interfering with enumext nested in keyans* (yes, its possible and crayze)
3297      \bool_set_false:N \l__enumext_store_active_bool
3298      \int_compare:nNnT { \l__enumext_level_int } > { 1 }
3299        {
3300          \msg_error:nn { enumext } { keyans-wrong-level }
3301        }
3302    }
```

(*End of definition for* __enumext_safe_exec_viii:.)

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

```
3303  \cs_new_protected:Npn \__enumext_parse_keys_viii:n #1
3304    {
3305      \tl_if_novalue:nF {#1}
3306        {
3307          \keys_set:nn { enumext / keyans* } {#1}
3308        }
3309    }
```

(*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.

```
3310  \cs_new_protected:Nn \__enumext_before_list_viii:
3311    {
3312      \__enumext_vspace_above_viii:
3313      \__enumext_before_args_exec_viii:
3314      \__enumext_start_mini_viii:
3315    }
```

(*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.

```
3316  \cs_new_protected:Nn \__enumext_after_list_viii:
3317    {
```

```
3318        \__enumext_stop_mini_viii:
3319        \__enumext_after_stop_list_viii:
3320        \__enumext_vspace_below_viii:
3321      }
```

(*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.

```
3322  \cs_new_protected_nopar:Nn \__enumext_start_item_tmp_viii:
3323    {
3324      \__enumext_stop_item_tmp_viii:
3325      \int_incr:N \l__enumext_item_column_pos_viii_int
3326      \int_gincr:N \g__enumext_item_count_all_viii_int
3327      \__enumext_item_peek_args_viii:
3328    }
```

(*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).

```
3329  \cs_new_protected:Nn \__enumext_item_peek_args_viii:
3330    {
3331      \peek_meaning:NTF (
3332        { \__enumext_joined_item_viii:w }
3333        { \__enumext_joined_item_viii:w (1) }
3334    }
```

(*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.

```
3335  \cs_new_protected:Npn \__enumext_joined_item_viii:w (#1)
3336    {
3337      \__enumext_starred_joined_item_viii:n {#1}
3338      \peek_meaning_remove:NTF *
3339        { \__enumext_starred_item_viii:w  }
3340        { \__enumext_standard_item_viii:w }
3341    }
```

(*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].

```
3342  \cs_new_protected:Npn \__enumext_standard_item_viii:w
3343    {
3344      \bool_set_false:N \l__enumext_item_starred_viii_bool
3345        \peek_meaning:NTF [
3346          {
```

```
3347            \bool_set_eq:NN
3348              \l__enumext_wrap_label_viii_bool
3349              \l__enumext_wrap_label_opt_viii_bool
3350            \__enumext_start_item_viii:w
3351          }
3352          {
3353            \bool_set_true:N \l__enumext_wrap_label_viii_bool
3354            \legacy_if_set_true:n { @noitemarg }
3355            \__enumext_start_item_viii:w [ \l__enumext_label_viii_tl ]
3356          }
3357      }
```

(*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*⟩`]`.

```
3358  \cs_new_protected:Npn \__enumext_starred_item_viii:w
3359    {
3360      \bool_set_true:N \l__enumext_item_starred_viii_bool
3361      \bool_set_true:N \l__enumext_wrap_label_viii_bool
3362      \peek_meaning:NTF [
3363        { \__enumext_starred_item_viii_aux_i:w }
3364        { \__enumext_starred_item_viii_aux_ii:w }
3365    }
```

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*⟩.

```
3366  \cs_new_protected:Npn \__enumext_starred_item_viii_aux_i:w [#1]
3367    {
3368      \tl_clear:N \l__enumext_store_keyans_label_tl
3369      \tl_if_novalue:nF { #1 }
3370        {
3371          \tl_set:Ne \l__enumext_keyans_tmpa_tl { \c_space_tl [#1] }
3372          \tl_set:Ne \l__enumext_keyans_tmpb_tl { \c_space_tl #1 }
3373        }
3374      \__enumext_starred_item_viii_aux_ii:w
3375    }
3376  \cs_new_protected:Npn \__enumext_starred_item_viii_aux_ii:w
3377    {
3378      \legacy_if_set_true:n { @noitemarg }
3379      \__enumext_start_item_viii:w [ \l__enumext_label_viii_tl ]
3380    }
```

(*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 store-ref are implemented.

```
3381  \cs_new_protected:Nn \__enumext_starred_item_exec:
3382    {
3383      \tl_put_left:Ne \l__enumext_store_keyans_label_tl { \l__enumext_label_viii_tl }
3384      \tl_if_blank:VF \l__enumext_keyans_tmpb_tl
3385        {
3386          \tl_put_right:Ne \l__enumext_store_keyans_label_tl { \l__enumext_keyans_tmpb_tl }
3387        }
3388      \__enumext_store_addto_prop:V \l__enumext_store_keyans_label_tl
3389      \__enumext_keyans_store_ref:
3390      \tl_put_left:Ne \l__enumext_store_keyans_label_tl { \item }
3391      \__enumext_keyans_addto_seq_link:
3392      \bool_if:NT \l__enumext_show_answer_bool
3393        {
3394          \tl_if_blank:VF \l__enumext_keyans_tmpa_tl
3395            {
3396              \tl_put_right:Ne \l__enumext_label_viii_tl { \l__enumext_keyans_tmpa_tl }
3397              \__enumext_label_width_by_box:Nn \l__enumext_keyans_tmpa_dim { \l__enumext_keyans_tmpa
3398              \dim_add:Nn \l__enumext_fake_item_indent_viii_dim { \l__enumext_keyans_tmpa_dim }
3399            }
3400          \__enumext_print_keyans_box:NN \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
```

```
3401              }
3402          \bool_if:NT \l__enumext_show_position_bool
3403            {
3404              \tl_set:Ne \l__enumext_mark_answer_sym_tl
3405                {
3406                  \group_begin:
3407                    \exp_not:N \normalfont
3408                    \exp_not:N \footnotesize [ \int_eval:n
3409                      {
3410                        \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
3411                      }
3412                    ]
3413                  \group_end:
3414                }
3415              \tl_if_blank:VF \l__enumext_keyans_tmpa_tl
3416                {
3417                  \tl_put_right:Ne \l__enumext_label_viii_tl { \l__enumext_keyans_tmpa_tl }
3418                  \__enumext_label_width_by_box:Nn \l__enumext_keyans_tmpa_dim { \l__enumext_keyans_tmpa
3419                  \dim_add:Nn \l__enumext_fake_item_indent_viii_dim { \l__enumext_keyans_tmpa_dim }
3420                }
3421              \__enumext_print_keyans_box:NN \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
3422          }
3423      }
```

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

Real definition of `\item` **in keyans***

The implementation at this point is very similar to that of the enumext* environment.

`\__enumext_start_item_viii:w`

```
3424  \cs_new_protected_nopar:Npn \__enumext_start_item_viii:w [#1]
3425    {
3426      \cs_set_eq:NN \__enumext_stop_item_tmp_viii: \__enumext_stop_item_viii:
3427      \legacy_if:nT { @noitemarg }
3428        {
3429          \legacy_if_set_false:n { @noitemarg }
3430          \legacy_if:nT { @nmbrlist }
3431            {
3432              \bool_if:NT \l__enumext_hyperref_bool
3433                {
3434                  \legacy_if_set_true:n { @hyper@item }
3435                }
3436              \refstepcounter{enumXviii}
3437            }
3438        }
```

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

```
3439          \group_begin:
3440            \lrbox{ \l__enumext_item_text_viii_box }
3441              \bool_if:NF \l__enumext_footnotes_key_bool
3442                {
3443                  \__enumext_renew_footnote:
3444                }
3445              \bool_if:NT \l__enumext_item_starred_viii_bool
3446                {
3447                  \__enumext_starred_item_exec:
3448                }
3449              \group_begin:
3450                \tl_use:N \l__enumext_label_font_style_viii_tl
3451                \bool_if:NTF \l__enumext_wrap_label_viii_bool
3452                  {
3453                    \makebox[ \l__enumext_labelwidth_viii_dim ][ \l__enumext_align_label_viii_str ]
3454                      { \__enumext_wrapper_label_viii:n {#1} }
3455                  }
3456                  {
3457                    \makebox[ \l__enumext_labelwidth_viii_dim ][ \l__enumext_align_label_viii_str ]{ #1
3458                  }
3459              \group_end:
3460              \skip_horizontal:N \l__enumext_labelsep_viii_dim
3461              \tl_use:N \l__enumext_after_list_args_viii_tl
3462              \__enumext_minipage:w [ t ]{ \l__enumext_joined_width_viii_dim  }
3463                \skip_set_eq:NN \parindent \l__enumext_listparindent_viii_dim
```

```
3464              \skip_set_eq:NN \parskip \l__enumext_parsep_viii_skip
3465              \tl_use:N \l__enumext_fake_item_indent_viii_tl
3466          }
```

(*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.

```
3467  \cs_new_protected_nopar:Nn \__enumext_stop_item_viii:
3468    {
3469         \__enumext_endminipage:
3470       \endlrbox
3471     \group_end:
3472     \box_set_wd:Nn \l__enumext_item_text_viii_box
3473       {
3474         \l__enumext_joined_width_viii_dim
3475         + \l__enumext_labelwidth_viii_dim
3476         + \l__enumext_labelsep_viii_dim
3477       }
3478     \int_set:Nn \hbadness { 10000 }
3479     \box_use:N \l__enumext_item_text_viii_box
3480     \bool_if:NF \l__enumext_footnotes_key_bool
3481       {
3482         \__enumext_print_footnote:
3483       }
3484     \int_compare:nNnTF { \l__enumext_item_column_pos_viii_int } = { \l__enumext_columns_viii_int }
3485       {
3486         \par\noindent
3487         \int_zero:N \l__enumext_item_column_pos_viii_int
3488       }
3489       { \hspace{ \l__enumext_columns_sep_viii_dim } }
3490    }
```

(*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.

```
3491  \cs_new_protected:Nn \__enumext_remove_extra_parsep_viii:
3492    {
3493     \int_compare:nNnT
3494       {
3495         \int_mod:nn {  \g__enumext_item_count_all_viii_int } { \l__enumext_columns_viii_int }
3496       }
3497       =
3498       { \c_zero_int }
3499       {
3500         \par
3501         \vspace{ -\l__enumext_itemsep_viii_skip }
3502         \int_gzero:N \g__enumext_item_count_all_viii_int
3503       }
3504    }
```

(*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.

```
3505  \NewDocumentCommand \getkeyans { m }
3506    {
3507     \exp_args:Ne \__enumext_getkeyans_aux:n
3508       { \tl_to_str:e { \text_expand:n {#1} } }
3509    }
```

(*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.

```
3510  \cs_new_protected:Npn \__enumext_getkeyans_aux:n #1
3511    {
```

```
3512      \str_if_in:nnTF {#1} { : }
3513        {
3514          \use:e
3515            {
3516              \cs_set:Npn \exp_not:N \__enumext_tmp:w ##1 \c_colon_str ##2 \scan_stop:
3517                { {##1} {##2} }
3518            }
3519          \exp_after:wN \__enumext_getkeyans:nn \__enumext_tmp:w #1 \scan_stop:
3520        }
3521        { \msg_error:nnn { enumext } { missing-colon } {#1} }
3522    }
```

(*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*⟩.

```
3523  \cs_new_protected:Npn \__enumext_getkeyans:nn #1 #2
3524    {
3525      \prop_if_exist:cF { g__enumext_#1_prop }
3526        { \msg_error:nnn { enumext } { undefined-storage-anskey } {#1} }
3527      \group_begin:
3528        \prop_item:cn { g__enumext_#1_prop }{#2}
3529      \group_end:
3530    }
```

(*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`.

```
3531  \keys_define:nn  { keyanskey / print }
3532    {
3533      level-1 .code:n     = \tl_put_right:Nn \l__enumext_print_keyans_i_tl
3534                            {
3535                              \setenumext[level,1] {#1} \setenumext[print,1] {#1}
3536                            },
3537      level-1 .initial:n = { label=\arabic*., nosep, columns=2, first=\small, font=\small },
3538      level-2 .code:n     = \tl_put_right:Nn \l__enumext_print_keyans_ii_tl
3539                            {
3540                              \setenumext[level,2] {#1} \setenumext[print,2] {#1}
3541                            },
3542      level-2 .initial:n = { nosep, label=(\alph*), first=\small, font=\small },
3543      level-3 .code:n     = \tl_put_right:Nn \l__enumext_print_keyans_iii_tl
3544                            {
3545                              \setenumext[level,3] {#1} \setenumext[print,3] {#1}
3546                            },
3547      level-3 .initial:n = { nosep, label=\roman*., first=\small, font=\small },
3548      level-4 .code:n     = \tl_put_right:Nn \l__enumext_print_keyans_iv_tl
3549                            {
3550                              \setenumext[level,4] {#1} \setenumext[print,4] {#1}
3551                            },
3552      level-4 .initial:n = { nosep, label=\Alph*., first=\small, font=\small },
3553      level-* .code:n     = \tl_put_right:Nn \l__enumext_print_keyans_vii_tl % starred
3554                            {
3555                              \setenumext[enumext*] {#1} %%\setenumext[print,*] {#1}
3556                            },
3557      level-* .initial:n = { label=\arabic*., nosep, columns=2, first=\small, font=\small },
3558    }
```

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

```
3559  \NewDocumentCommand \printkeyans { s O{} m }
3560    {
3561      \group_begin:
3562        \tl_use:N \l__enumext_print_keyans_i_tl
3563        \tl_use:N \l__enumext_print_keyans_ii_tl
3564        \tl_use:N \l__enumext_print_keyans_iii_tl
3565        \tl_use:N \l__enumext_print_keyans_iv_tl
```

```
3566        \tl_use:N \l__enumext_print_keyans_vii_tl
3567        \__enumext_printkeyans:nnn { #1 } { #2 } { #3 }
3568      \group_end:
3569    }
```

(*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

```
3570 \cs_new_protected:Npn \__enumext_printkeyans:nnn #1 #2 #3
3571    {
3572      \seq_if_exist:cTF { g__enumext_#3_seq }
3573        {
3574          \seq_if_empty:cF { g__enumext_#3_seq }
3575            {
3576              %%\seq_show:c { g__enumext_#3_seq }
3577              \bool_if:nTF {#1}
3578                {
3579                  \begin{enumext*}[#2]
3580                    \seq_map_inline:cn { g__enumext_#3_seq } { ##1 }
3581                  \end{enumext*}
3582                }
3583                {
3584                  \begin{enumext}[#2]
3585                    \seq_map_inline:cn { g__enumext_#3_seq } { ##1 }
3586                  \end{enumext}
3587                }
3588            }
3589        }
3590        {
3591          \msg_error:nnn { enumext } { undefined-storage-anskey } {#3}
3592        }
3593    }
```

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

10.39   The command \setenumext

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

```
3594 \keys_define:nn { enumext / meta-families }
3595    {
3596      level-1  .code:n = { \keys_set:nn { enumext / level-1  } {#1} } ,
3597      level-2  .code:n = { \keys_set:nn { enumext / level-2  } {#1} } ,
3598      level-3  .code:n = { \keys_set:nn { enumext / level-3  } {#1} } ,
3599      level-4  .code:n = { \keys_set:nn { enumext / level-4  } {#1} } ,
3600      keyans   .code:n = { \keys_set:nn { enumext / keyans   } {#1} } ,
3601      enumext* .code:n = { \keys_set:nn { enumext / enumext* } {#1} } ,
3602      keyans*  .code:n = { \keys_set:nn { enumext / keyans*  } {#1} } ,
3603      print-1  .code:n = { \keys_set:nn { keyanskey / print } { level-1 = {#1} } } ,
3604      print-2  .code:n = { \keys_set:nn { keyanskey / print } { level-2 = {#1} } } ,
3605      print-3  .code:n = { \keys_set:nn { keyanskey / print } { level-3 = {#1} } } ,
3606      print-4  .code:n = { \keys_set:nn { keyanskey / print } { level-4 = {#1} } } ,
3607      print-*  .code:n = { \keys_set:nn { keyanskey / print } { level-* = {#1} } } ,
3608      unknown  .code:n = { \msg_error:nn { enumext } { unknown-key-family } } ,
3609    }
```

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

```
3610 \seq_const_from_clist:Nn \c__enumext_all_families_seq
3611    {
3612      level-1 , level-2 , level-3 , level-4 , keyans, enumext*,
3613      keyans* , print-1 , print-2 , print-3 , print-4 , print-*,
3614    }
```

\setenumext   Now we define the user command \setenumext.

```
3615 \NewDocumentCommand \setenumext { o +m }
3616    {
3617      \tl_if_novalue:nTF {#1}
```

```
3618        {
3619            \seq_map_inline:Nn \c__enumext_all_families_seq
3620        }
3621        {
3622            \seq_clear:N \l__enumext_setkey_tmpa_seq
3623            \seq_set_from_clist:Nn \l__enumext_setkey_tmpb_seq {#1}
3624            \int_set:Nn \l__enumext_setkey_tmpa_int
3625              {
3626                \seq_count:N \l__enumext_setkey_tmpb_seq
3627              }
3628            \int_compare:nNnTF { \l__enumext_setkey_tmpa_int } > { 1 }
3629              {
3630                \seq_pop_left:NN \l__enumext_setkey_tmpb_seq \l__enumext_setkey_tmpa_tl
3631                \seq_map_function:NN \l__enumext_setkey_tmpb_seq \__enumext_set_parse:n
3632                \seq_set_map_e:NNn \l__enumext_setkey_tmpa_seq \l__enumext_setkey_tmpa_seq
3633                  {
3634                    \tl_use:N \l__enumext_setkey_tmpa_tl - ##1
3635                  }
3636              }
3637              {
3638                \seq_put_right:Ne \l__enumext_setkey_tmpa_seq { \tl_trim_spaces:n {#1} }
3639              }
3640            \seq_if_empty:NTF \l__enumext_setkey_tmpa_seq
3641              { \seq_map_inline:Nn \c__enumext_all_families_seq }
3642              { \seq_map_inline:Nn \l__enumext_setkey_tmpa_seq }
3643        }
3644        {
3645            \keys_set:nn { enumext / meta-families } { ##1 = {#2} }
3646        }
3647    }
```

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

__enumext_set_parse:n   Internal functions used by the \setenumext command.
__enumext_set_error:nn

```
3648  \cs_new_protected:Npn \__enumext_set_parse:n #1
3649    {
3650      \tl_set:Ne \l__enumext_setkey_tmpb_tl { \tl_trim_spaces:n {#1} }
3651      \int_step_inline:nnn { 0 } { 4 } % <- max level
3652        { \tl_remove_all:Nn \l__enumext_setkey_tmpb_tl {##1} }
3653      \tl_if_empty:NTF \l__enumext_setkey_tmpb_tl
3654        {
3655          \seq_put_right:Ne \l__enumext_setkey_tmpa_seq
3656            { \tl_trim_spaces:n {#1} }
3657        }
3658        { \__enumext_set_error:nn {#1} { } }
3659    }
3660  \cs_new_protected:Npn \__enumext_set_error:nn #1 #2
3661    { \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.

```
3662  \msg_new:nnn { enumext } { package-load }
3663    {
3664      The ~ '#1' ~ package ~ is ~ already ~ loaded.
3665    }

3666  \msg_new:nnn { enumext } { package-not-load }
3667    {
3668      The ~ '#1' ~ package ~ will ~ be ~ loaded ~ as ~ a ~ dependency.
3669    }

3670  \msg_new:nnn { enumext } { package-load-foot }
3671    {
3672      The ~ '#1' ~ package ~ is ~ loaded ~ with ~ the ~ option ~ '#2'.
3673    }
```

Message used in the creation of counters by enumext package.

```
3674  \msg_new:nnn { enumext } { counters }
3675    {
3676      The ~ counter ~ '#1' ~ is ~ already ~ defined ~ by ~ some ~ \\
```

```
3677        package ~ or ~ macro, ~ it ~ cannot ~ be ~ continued.
3678      }
```

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

```
3679  \msg_new:nnn { enumext } { invalid-key }
3680    {
3681      The ~ key ~ '#1' ~ is ~ not ~ know ~ the ~ level ~ #2.
3682    }
3683  \msg_new:nnn { enumext } { unknown-key-family }
3684    {
3685      Unknown~key~family~`\l_keys_key_str'~for~enumext.
3686    }
```

Messages used in length calculation.

```
3687  \msg_new:nnn { enumext } { width-negative }
3688    {
3689      Ignoring ~ negative ~ value ~ '#1=#2' ~ \msg_line_context:.\\
3690      The ~ key ~ '#1'~ accepts ~ values  ~ >= ~ 0pt.
3691    }
3692  \msg_new:nnn { enumext } { width-zero }
3693    {
3694      Invalid ~ '#1=#2' ~ \msg_line_context:.\\
3695      The ~ key ~ '#1'~ accepts ~ values  ~ > ~ 0pt.
3696    }
```

Messages used by show-length key in enumext.

```
3697  \msg_new:nnn { enumext } { list-lengths }
3698    {
3699      **** ~ Lengths ~ used ~ by ~ 'enumext' ~ level ~ '#2' ~ \msg_line_context:~\c_space_tl ****\\
3700      \__enumext_show_length:nnn { dim  } { labelsep     } {#1}
3701      \__enumext_show_length:nnn { dim  } { labelwidth   } {#1}
3702      \__enumext_show_length:nnn { dim  } { itemindent   } {#1}
3703      \__enumext_show_length:nnn { dim  } { leftmargin   } {#1}
3704      \__enumext_show_length:nnn { dim  } { rightmargin  } {#1}
3705      \__enumext_show_length:nnn { dim  } { listparindent } {#1}
3706      \__enumext_show_length:nnn { skip } { topsep     } {#1}
3707      \__enumext_show_length:nnn { skip } { parsep     } {#1}
3708      \__enumext_show_length:nnn { skip } { partopsep } {#1}
3709      \__enumext_show_length:nnn { skip } { itemsep    } {#1}
3710      **************************************************
3711    }
```

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

```
3712  \msg_new:nnn { enumext } { list-lengths-not-nested }
3713    {
3714      **** ~ Lengths ~ used ~ by ~ '#2' ~ environment ~ \msg_line_context:~\c_space_tl ****\\
3715      \__enumext_show_length:nnn { dim  } { labelsep     } {#1}
3716      \__enumext_show_length:nnn { dim  } { labelwidth   } {#1}
3717      \__enumext_show_length:nnn { dim  } { itemindent   } {#1}
3718      \__enumext_show_length:nnn { dim  } { leftmargin   } {#1}
3719      \__enumext_show_length:nnn { dim  } { rightmargin  } {#1}
3720      \__enumext_show_length:nnn { dim  } { listparindent } {#1}
3721      \__enumext_show_length:nnn { skip } { topsep     } {#1}
3722      \__enumext_show_length:nnn { skip } { parsep     } {#1}
3723      \__enumext_show_length:nnn { skip } { partopsep } {#1}
3724      \__enumext_show_length:nnn { skip } { itemsep    } {#1}
3725      **************************************************
3726    }
```

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

```
3727  \msg_new:nnn { enumext } { items-same-answer }
3728    {
3729      **********~Checking~answers~on~'#1'~OK~*********\\
3730      **~ All ~ items ~ stored ~ in ~ sequence ~ '#1' ~ have ~ an ~ answer. \\
3731      *************************************
3732      \prg_replicate:nn { 7 + \str_count:n {#1} } { * }
3733    }
3734  \msg_new:nnn { enumext } { item-different-answer }
3735    {
3736      Number ~ of ~ items ~ different ~  of ~ number ~ of ~
3737      answer ~ in ~ sequence ~ '#1'~ closed ~ \msg_line_context:.
3738    }
```

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

```
3739 \msg_new:nnn { enumext } { missing-starred }
3740   {
3741     Missing ~ '\c_backslash_str #1*' ~ in ~ '#2' ~ \msg_line_context:.
3742   }
```

Message for the nesting depth of the environment `enumext`.

```
3743 \msg_new:nnn { enumext } { list-too-deep }
3744   {
3745     Too ~ deep ~ nesting  ~ for ~ 'enumext' ~ \msg_line_context:.~ \\
3746     The ~ maximum  ~ level  ~ of  ~ nesting  ~ is ~ 4.
3747   }
```

Messages used by `\anskey` and `\anspic` commands.

```
3748 \msg_new:nnn { enumext } { anskey-wrong-place }
3749   {
3750     Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:.~ \\
3751     '\c_backslash_str #1' ~ works ~ in ~ the ~ environment ~ '#2'.
3752   }
3753 \msg_new:nnn { enumext } { anspic-wrong-place }
3754   {
3755     Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:.~ \\
3756     '\c_backslash_str #1' ~ works ~ in ~ the ~ environment ~ '#2'.
3757   }
3758 \msg_new:nnn { enumext } { command-wrong-place }
3759   {
3760     Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:.~ \\
3761     '\c_backslash_str #1' ~ works ~ outside ~ the ~ environment ~ '#2'.
3762   }
```

Messages used by `keyans` and `keyanspic` environment.

```
3763 \msg_new:nnn { enumext } { keyans-nested }
3764   {
3765     The ~ environment ~ 'keyans' ~ can't ~ be  ~ nested  ~ \msg_line_context:.
3766   }
3767 \msg_new:nnn { enumext } { keyans-wrong-level }
3768   {
3769     Wrong ~ level ~ position ~ for ~ 'keyans' ~ \msg_line_context:.~ \\
3770     The ~ environment ~ 'keyans' ~ can ~ only ~ be ~ in ~ the ~ first ~ level.
3771   }
3772 \msg_new:nnn { enumext } { wrong-place }
3773   {
3774     Wrong ~ place ~ for ~ '#1' ~ environment ~\msg_line_context:.~ \\
3775     '#1' ~ is ~ only ~ found ~ with ~ '#2' ~  in  ~  'enumext.
3776   }
3777 \msg_new:nnn { enumext } { keyanspic-nested }
3778   {
3779     The ~ environment ~ 'keyanspic' ~ can't ~ be  ~ nested~ \msg_line_context:.~.
3780   }
3781 \msg_new:nnn { enumext } { keyanspic-wrong-level }
3782   {
3783     Wrong ~ level ~ position ~ for ~ 'keyanspic' ~ \msg_line_context:.~ \\
3784     The ~ environment ~ 'keyans' ~ can ~ only ~ be ~ in ~ the ~ first ~ level.
3785   }
```

Messages used by `\getkeyans` command.

```
3786 \msg_new:nnn { enumext } { undefined-storage-anskey }
3787   {
3788     Storage ~ named ~ '#1' ~ is ~ not ~ defined ~ \msg_line_context:.
3789   }
```

Messages used by `\miniright` command.

```
3790 \msg_new:nnn { enumext } { missing-miniright }
3791   {
3792     Missing ~ '\c_backslash_str miniright' ~ in ~ \msg_line_context:.\\
3793     The ~ key ~ 'mini-env' ~ need ~ '\c_backslash_str miniright'.
3794   }
3795 \msg_new:nnn { enumext } { wrong-miniright-place }
3796   {
3797     Wrong ~ place ~ for ~ '\c_backslash_str miniright' ~ \msg_line_context:.~ \\
3798     Works ~ in ~ 'enumext' ~ and ~ 'keyans' ~ with ~ key ~ 'mini-env'.
3799   }
3800 \msg_new:nnn { enumext } { wrong-miniright-use }
```

```
3801    {
3802      Wrong ~ use ~ for ~ '\c_backslash_str miniright' ~ \msg_line_context:.~ \\
3803      '\c_backslash_str miniright' ~ need ~ a ~ key ~ 'mini-env'.
3804    }
```

Messages used by enumext* and keyans* environments.

```
3805  \msg_new:nnn { enumext } { nested }
3806    {
3807      The ~ starred ~ environment ~ can't ~ be ~ nested ~ \msg_line_context:.
3808    }
3809  \msg_new:nnn { enumext } { item-joined }
3810    {
3811      Items ~ joined ~ (#1) ~ > ~ #2  ~ columns ~\msg_line_context:.
3812    }
3813  \msg_new:nnn { enumext } { item-joined-columns }
3814    {
3815      Not ~ space ~ to ~ join ~ items ~ (#1) ~ > ~ #2 ~\msg_line_context:.
3816    }
```

10.41   Finish package

Finish package implementation.

```
3817  \file_input_stop:
3818  ⟨/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.