enumext
ENUMERATE EXERCISE SHEETS

V1.0   2024-05-12[*]

©2024 by Pablo González[†]

Abstract

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

Contents

Motivation and acknowledgments

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

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

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

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

---

[*]This file describes a documentation for v1.0, last revised 2024-05-12.
[†]E-mail: «pablgonz@educarchile.cl».

License and Requirements

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

The enumext package loads and requires `multicol`[3] package, need to have a modern TeX distribution such as TeX Live or MiKTeX. It has been tested with the standard classes provided by LaTeX: `book`, `report`, `article` and `letter` on `10pt`, `11pt` and `12pt`.

1 Introduction

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

4. Question with image and label below:

(A)   (B)   (C)

(D)   (E)

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

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

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

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

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

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

1.1 Description and usage

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

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

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

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

The package is loaded in the usual way:

```
\usepackage{enumext}
```

1.2 The concept of left margin

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

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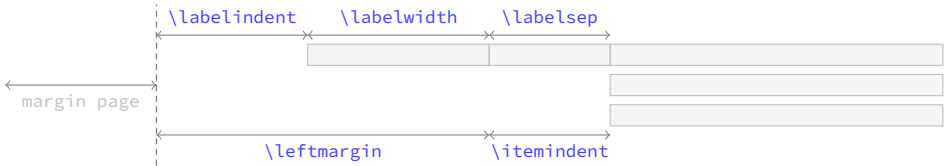

Figure 1: Representation of horizontal lengths in enumitem.

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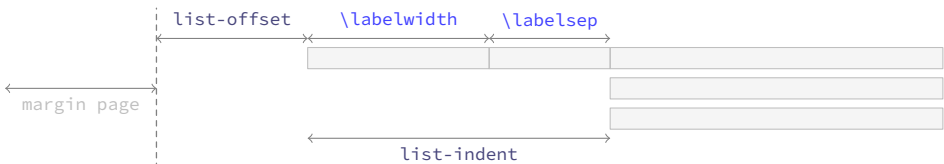
Figure 2: Representation of horizontal lengths concept in enumext.

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

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

1.3 User interface

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

1.3.1 Internal counters

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

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

1.3.2 Support for multicol

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

Figure 4: Representation of the two column output for a nested level in `enumext` environment.
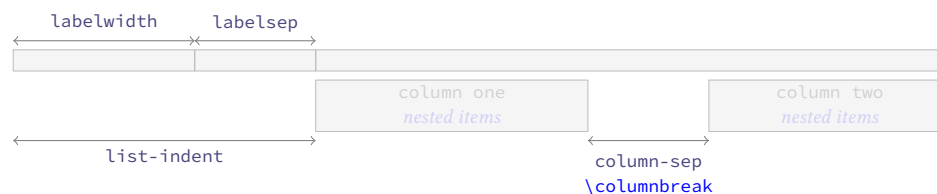
The *"non starred"* version of the `multicols` environment is always used together with the `\raggedcolumns` command and is controlled by `columns` and `columns-sep` keys. The environment is available for all nesting levels, and can can together with the `mini-env` key. If you need to force a start a new column `\columnbreak` must be used (see §3.5).

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

1.3.3 Support for `minipage`

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

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

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

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

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

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

1.3.5 Support for `\footnote`

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

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

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

2 The environment enumext

enumext
enumext*

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

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

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

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

Example

1. This text is in the first level.

    (a) This text is in the second level.

        i. This text is in the third level.

            A. This text is in the fourth level.

    X This text is in the first level.

⋆ 2. This text is in the first level.

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

2.1   The \item* in enumext

\item*

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

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

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

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

2.1.1   Keys for \item* in enumext

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

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

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

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

3   The command \setenumext

\setenumext

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

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

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

💣* It should be kept in mind that using any ⟨*key*⟩ that sets a *rubber 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 `save-ref` key. You cannot use commands with ⟨*label*⟩ as an argument, for example `\emph{`⟨`\alph*`⟩`}` will return an error. For full customization of how ⟨*label*⟩ is displayed use the `font` or `wrap-label` keys.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

3.2 Keys for spaces

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

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

3.2.1   Vertical spaces

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

3.2.2   Horizontal spaces

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

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

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

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

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

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

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

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

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

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

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

3.3 Keys for add code

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

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

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

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

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

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

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

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

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

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

3.4 Keys for `start` 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.

3.6.2 The key miniright

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

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

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

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

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

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` and `keyans*` environments and `\anspic*` in `keyanspic` environment. If the ⟨*sequence*⟩ or ⟨*prop list*⟩ does not exist, it will be created globally.

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

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

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

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

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

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

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

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

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

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

4.2   Keys for internal `label` and `ref`

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

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

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

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

4.3   Keys for debugging and checking

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

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

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

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

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

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

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

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

4.4   The command `\anskey`

`\anskey` `\anskey{`⟨*content*⟩`}`

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

Example

⋆ 1. Text containing our instructions or questions.

　　* first answer

2. Text containing our instructions or questions.

　　(a) Question.

　　　* second answer

3. Text containing our instructions or questions.

　　* third answer

4. Text containing our instructions or questions.

　　* fourth answer

```
\begin{enumext}[save-ans=test,show-ans=true]
  \item* Text containing our instructions or questions. \anskey{⟨first answer⟩}
  \item Text containing our instructions or questions.
    \begin{enumext}
      \item Question.\anskey{⟨second answer⟩}
    \end{enumext}
  \item Text containing our instructions or questions. \anskey{⟨third answer⟩}
  \item Text containing our instructions or questions. \anskey{⟨fourth answer⟩}
\end{enumext}
```

4.5   The environment `keyans`

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

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

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

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

4.5.1 The \item* in keyans

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

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

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

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

Example

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

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

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

1. Text containing a question.

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

2. Text containing a question and image.

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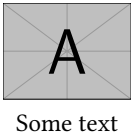
Some text

4.6 The environment keyanspic

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

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

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.

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

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}
```
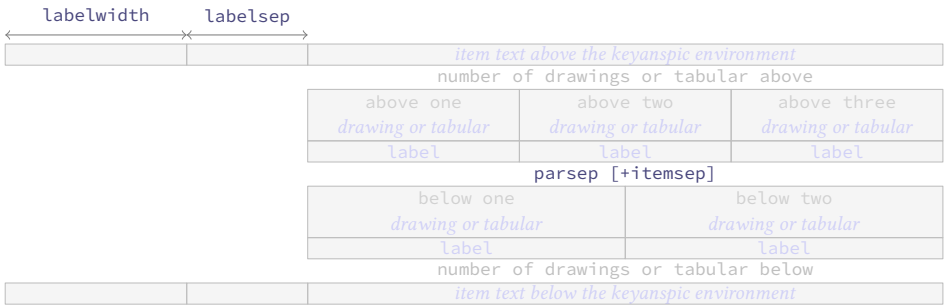
1. Question with images.
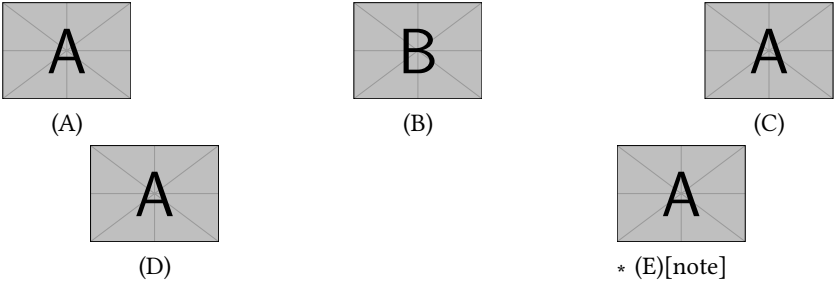
4.7 Printing stored content

4.7.1 The command \getkeyans

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

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

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

4.7.2 The command \printkeyans

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

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

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

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

Example

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

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

  \item Related to Linux

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

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

\printkeyans{sample}
```

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

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

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

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

5 Full examples

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

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

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

Example 1

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

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

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

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

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

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

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

---

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

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

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

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

Example 2

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

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

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

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

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

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

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

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

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

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

Example 3

A *"simple multiple choice"* test 📄.

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

   Ⓐ

   Ⓑ

   Ⓒ

   Ⓓ

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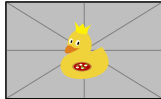
5. Question with image on left side:
   - Ⓐ value
   - Ⓑ value
   - Ⓒ value
   - Ⓓ correct
   - Ⓔ value

Test keys

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

Example 4

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

🦆 1. Factor $x^2 - 2x + 1$

🦆 2. Factor $3x + 3y + 3z$

The following questions need to be cuaqtified :)

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

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

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

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

Example 5

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

1

Which choice best describes what happens in the passage?

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

2

Which choice best describes what happens in the passage?

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

3

Which choice best describes what happens in the passage?

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

4

Which choice best describes what happens in the passage?

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

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

6   The way of non-enumerated lists

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

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

Fake `itemize` environment

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

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

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

Fake `description` environment

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

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

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

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

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

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

Description indented by label

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

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

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

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

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

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

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

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

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

Description with multi-line labels

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

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

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

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

Final notes

The original implementation (if you can call it that) of the ideas that led to the creation of enumext were some macros using the enumerate[4] package for personal use created in early 2003, the code was quite questionable, but functional for these simple requirements.

With the great answers given by Christian Hupfer in Create a fake label ref using list and the answer given by David Carlisle in Change the use of label ref by data save in an array (list) I managed to create a more solid code than the original version, now using the `l3prop`[10] and `l3seq`[10] modules together with the `hyperref`[7] and `enumitem`[5] packages, which did the job, but with some limitations.

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

Why list environments?

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

Why not random questions and other utilities

The *"random"* type questions I love and hate them at the same time, although they simplify a lot the work when creating a multiple choice test, but you lose the beauty of typesetting a document with LaTeX, that is to say the output does not always look as nice as it should, even if they are only alternatives these must follow a certain order when presented either numerical or presentation, that said handling that using *nested lists* is quite complicated so I do not classify to be implemented.

7   References

[1] Hirschhorn, Philip. "Using the exam document class". Available from CTAN, https://www.ctan.org/pkg/exam, 2023.

[2] Niederberger, Clemens. "xsim – eXercise Sheets IMproved". Available from CTAN, https://www.ctan.org/pkg/xsim, 2023.

[3] Mittelbach, Frank. "An environment for multicolumn output". Available from CTAN, https://www.ctan.org/pkg/multicol, 2024.

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

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

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

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

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

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

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

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

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

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

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

8   Change history

v1.0 2024-05-12   – 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    {2024-05-12}
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.

\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

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

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

```
27 \bool_new:N \l__enumext_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`.)

```
\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.*)

```
\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.*)

```
\l__enumext_store_active_bool
\l__enumext_store_name_tl
\g__enumext_store_name_tl
\l__enumext_store_anskey_arg_tl
\l__enumext_store_columns_join_int
\l__enumext_store_keyans_label_tl
\l__enumext_store_keyans_item_opt_tl
\l__enumext_keyans_item_opt_tl
\l__enumext_keyans_tmpa_tl
\l__enumext_keyans_tmpb_tl
\l__enumext_keyans_tmpa_dim
```

The boolean variable `\l__enumext_store_active_bool` setting by `save-ans` key (§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_store_keyans_item_opt_tl
89  \tl_new:N   \l__enumext_keyans_item_opt_tl
90  \tl_new:N   \l__enumext_keyans_tmpa_tl
91  \tl_new:N   \l__enumext_keyans_tmpb_tl
92  \dim_new:N  \l__enumext_keyans_tmpa_dim
```

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

```
\l__enumext_setkey_tmpa_tl
\l__enumext_setkey_tmpb_tl
\l__enumext_setkey_tmpa_int
\l__enumext_setkey_tmpa_seq
\l__enumext_setkey_tmpb_seq
```

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

```
93  \tl_new:N  \l__enumext_setkey_tmpa_tl
94  \tl_new:N  \l__enumext_setkey_tmpb_tl
95  \int_new:N \l__enumext_setkey_tmpa_int
96  \seq_new:N \l__enumext_setkey_tmpa_seq
97  \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*.

```
98 \cs_set_protected:Npn \__enumext_tmp:n #1
99   {
100     \tl_new:c { l__enumext_store_opt_#1_tl            }
101     \tl_new:c { l__enumext_print_keyans_#1_tl         }
102     \bool_new:c { l__enumext_store_columns_#1_bool      }
103     \int_new:c { l__enumext_store_columns_#1_int        }
104     \bool_new:c { l__enumext_store_columns_sep_#1_bool }
105     \dim_new:c { l__enumext_store_columns_sep_#1_dim    }
106     \bool_new:c { l__enumext_store_upper_level_#1_bool }
107   }
108 \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).

```
109 \bool_new:N \l__enumext_show_answer_bool
110 \bool_new:N \l__enumext_show_position_bool
111 \tl_new:N    \l__enumext_mark_ref_sym_tl
112 \tl_new:N    \l__enumext_mark_answer_sym_tl
113 \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).

```
114 \seq_new:N  \l__enumext_keyans_pic_body_seq
115 \dim_new:N  \l__enumext_keyans_pic_width_dim
116 \int_new:N  \l__enumext_keyans_pic_above_int
117 \int_new:N  \l__enumext_keyans_pic_below_int
118 \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) controlled by the check-ans and no-store keys.

```
119 \bool_new:N \l__enumext_store_ans_bool
120 \bool_new:N \l__enumext_check_ans_bool
121 \bool_new:N \g__enumext_check_ans_show_bool
122 \bool_new:N \g__enumext_check_ans_show_h_bool
123 \tl_new:N    \g__enumext_check_ans_item_tl
124 \int_new:N  \g__enumext_count_item_anskey_int
125 \int_new:N  \g__enumext_count_item_number_int
126 \int_new:N  \g__enumext_standar_star_env_int
127 \int_new:N  \g__enumext_starred_star_env_int
128 \int_new:N  \g__enumext_starred_keyans_star_env_int
129 \int_new:N  \g__enumext_standar_keyans_star_env_int
130 \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.

```
131 \bool_new:N \l__enumext_hyperref_bool
132 \bool_new:N \l__enumext_footnotes_key_bool
```

(*End of definition for* \l__enumext_hyperref_bool *and* \l__enumext_footnotes_key_bool.)

\l__enumext_newlabel_arg_one_tl
\l__enumext_newlabel_arg_two_tl
\l__enumext_store_write_aux_file_tl
\l__enumext_label_copy_X_tl

Internal variables are used when executing the save-ref key. The variables \l__enumext_label_-copy_X_tl correspond to temporary copies of the labels defined by level on which operations will be performed.

The variables \l__enumext_newlabel_arg_one_tl and \l__enumext_newlabel_arg_two_tl will be used to form the arguments passed to the function __enumext_newlabel:nn and the variable \l__-enumext_store_write_aux_file_tl will be in charge of executing the writing code in the .aux file.

```
133 \tl_new:N \l__enumext_newlabel_arg_one_tl
134 \tl_new:N \l__enumext_newlabel_arg_two_tl
```

```
135 \tl_new:N \l__enumext_store_write_aux_file_tl
136 \cs_set_protected:Npn \__enumext_tmp:n #1
137    {
138       \tl_new:c { l__enumext_label_copy_#1_tl }
139    }
140 \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.

```
141 \int_new:N \g__enumext_footnote_int
142 \seq_new:N \g__enumext_footnote_arg_seq
143 \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).

```
144 \tl_const:Nn \c__enumext_counter_style_tl
145    { { arabic } { roman } { Roman } { alph } { Alph } }
146 \tl_new:N \l__enumext_ref_key_arg_tl
147 \tl_new:N \l__enumext_ref_aux_tl
148 \cs_set_protected:Npn \__enumext_tmp:n #1
149    {
150       \tl_new:c  { l__enumext_counter_style_for_ref_#1_tl }
151       \tl_new:c  { l__enumext_the_counter_#1_tl }
152       \tl_set:ce { l__enumext_the_counter_#1_tl } { \exp_not:c { theenumX#1 } }
153    }
154 \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.

```
155 \cs_set_protected:Npn \__enumext_tmp:n #1
156    {
157       \bool_new:c { l__enumext_item_starred_#1_bool    }
158       \int_new:c  { l__enumext_item_column_pos_#1_int  }
159       \int_new:c  { g__enumext_item_count_all_#1_int   }
160       \int_new:c  { l__enumext_joined_item_#1_int      }
161       \int_new:c  { l__enumext_joined_item_aux_#1_int  }
162       \int_new:c  { l__enumext_tmpa_#1_int             }
163       \box_new:c  { l__enumext_item_text_#1_box        }
164       \dim_new:c  { l__enumext_joined_width_#1_dim     }
165       \dim_new:c  { l__enumext_item_width_#1_dim       }
166       \tl_new:c   { g__enumext_item_symbol_aux_#1_tl   }
167       \str_new:c  { l__enumext_align_label_#1_str      }
168       \bool_new:c { g__enumext_minipage_active_#1_bool }
169       \tl_new:c   { g__enumext_miniright_code_#1_tl    }
170       \bool_new:c { g__enumext_minipage_center_#1_bool }
171       \dim_new:c  { g__enumext_minipage_right_#1_dim   }
172       \skip_new:c { g__enumext_minipage_right_#1_skip  }
173    }
174 \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.

```
175 \clist_const:Nn \c__enumext_all_envs_clist
176    {
177       {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv},
178       {keyans}{v}, {enumext*}{vii}, {keyans*}{viii}
179    }
```

(*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.

```
180 \cs_new_protected:Npn \__enumext_at_begin_document:n #1
181   {
182     \hook_gput_code:nnn {begindocument} {enumext} { #1 }
183   }
```

(*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.

```
184 \cs_new_protected:Npn \__enumext_after_env:nn #1 #2
185   {
186     \hook_gput_code:nnn {env/#1/after} {enumext} {#2}
187   }
```

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

`\__enumext_level:`

Function for check current level in `enumext`.

```
188 \cs_new:Nn \__enumext_level:
189   {
190     \int_to_roman:n { \l__enumext_level_int }
191   }
```

(*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?.

```
192 \prg_new_protected_conditional:Npnn \__enumext_if_is_int:n #1 { T, F, TF }
193   {
194     \regex_match:nnTF { ^[\+\-]?[\d]+$ } {#1} % $
195       { \prg_return_true: }
196       { \prg_return_false: }
197   }
```

(*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.

```
198 \cs_new:Npn \__enumext_show_length:nnn #1 #2 #3
199   {
200     * ~ #2
201     \prg_replicate:nn { 14 - \str_count:n {#2} } { ~ }
202       = ~ \use:c { #1_use:c } { l__enumext_#2_#3_#1 } \\
203   }
```

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

`\__enumext_zero_count_level:`

Internal function used by `check-ans` key.

```
204 \cs_set_protected:Nn \__enumext_zero_count_level:
205   {
206     \cs_set_protected:Npn \__enumext_tmp:n ##1
207       {
208         \int_gzero:c { g__enumext_count_level_##1_int }
209       }
210     \clist_map_inline:nn { i, ii, iii, iv, vii } { \__enumext_tmp:n {##1} }
211   }
```

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

`\__enumext_current_env:`

The function `\__enumext_current_env:` will set the global variables `\g__enumext_standar_bool` and `\g__enumext_starred_bool` with which we will distinguish whether the environments `enumext` and `enumext*` are nested in each other.

```
212 \cs_new_protected:Nn \__enumext_current_env:
213   {
214     \str_case:en { \@currenvir }
215       {
216         {enumext}
```

```
217                 {
218                     \int_compare:nNnT { \l__enumext_level_h_int } = { \c_zero_int }
219                         {
220                             \bool_gset_true:N \g__enumext_standar_bool
221                             \int_gset:Nn \g__enumext_standar_star_env_int { \inputlineno }
222                             \typeout{working-on-enumext}
223                         }
224                 }
225             {enumext*}
226                 {
227                     \int_compare:nNnT { \l__enumext_level_int } = { \c_zero_int }
228                         {
229                             \bool_gset_true:N \g__enumext_starred_bool
230                             \int_gset:Nn \g__enumext_starred_star_env_int { \inputlineno }
231                             \typeout{working-on-enumext*}
232                         }
233                 }
234         }
235     }
```

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

10.6   Copying `list` and `minipage` environments

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

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

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

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

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

```
236 \__enumext_at_begin_document:n
237     {
238         \cs_new_eq:NN \__enumext_start_list:nn \list
239         \cs_new_eq:NN \__enumext_stop_list: \endlist
240         \cs_new_eq:NN \__enumext_item_std:w \item
241     }
```

(*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.

```
242 \__enumext_at_begin_document:n
243     {
244         \cs_new_eq:NN \__enumext_minipage:w \minipage
245         \cs_new_eq:NN \__enumext_endminipage: \endminipage
246     }
```

(*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.

```
247 \hook_gput_code:nnn { begindocument } { enumext } { \__enumext_after_hyperref: }
248 \hook_gset_rule:nnnn { begindocument } { enumext } { after } { hyperref }
```

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

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

```
249 \cs_new_protected:Nn \__enumext_after_hyperref:
```

```
250    {
251      \IfPackageLoadedTF { hyperref }
252        {
253          \msg_info:nnn { enumext } { package-load } { hyperref }
254          \bool_set_true:N \l__enumext_hyperref_bool
255          \IfHyperBoolean{hyperfootnotes}
256            {
257              \typeout{hyperfootnotes=true}
258              \bool_set_true:N \l__enumext_footnotes_key_bool
259            }
260            { \typeout{hyperfootnotes=false} }
261        }
262        {  }
```

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.

```
263      \bool_if:NT \l__enumext_footnotes_key_bool
264        {
265          \IfPackageLoadedTF { footnotehyper }
266            {
267              \msg_info:nnn { enumext } { package-load } { footnotehyper }
268            }
269            {
270              \typeout{No ~ footnotehyper ~ load}
271              \typeout{Load ~ and  ~ use  ~ \string\makesavenoteenv{enumext*}}
272              \bool_set_false:N \l__enumext_footnotes_key_bool
273            }
274        }
```

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.

```
275      \bool_if:NTF \l__enumext_hyperref_bool
276        {
277          \cs_new_eq:NN \__enumext_hypertarget:nn \hypertarget
278          \cs_new_eq:NN \__enumext_phantomsection: \phantomsection
279        }
280        {
281          \cs_new_eq:NN \__enumext_hypertarget:nn \use_none:nn
282          \cs_new_eq:NN \__enumext_phantomsection: \prg_do_nothing:
283        }
284    }
```

(End of definition for __enumext_after_hyperref:*,* __enumext_hypertarget:nn*, and* __enumext_phantomsection:*.)*

__enumext_newlabel:nn The function __enumext_newlabel:nn write the information to the .aux file when using the save-ref key. The arguments taken by the function are:

#1: \l__enumext_newlabel_arg_one_tl

#2: \l__enumext_newlabel_arg_two_tl

💣 The trick here is to manage the number of arguments passed to \newlabel{#1}{#2} according to the presence of the hyperref package.

```
285 \cs_new_protected:Npn \__enumext_newlabel:nn #1 #2
286    {
287      \protected@write \@auxout { }
288        {
289          \token_to_str:N \newlabel {#1}
290            {
291              {#2}
292              \bool_if:NT \l__enumext_hyperref_bool
293                { { \thepage } {#2} {#1} }
294                { }
295            }
296        }
297      \__enumext_hypertarget:nn {#1} { }
298      \__enumext_phantomsection:
299    }
```

(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.

```
300 \cs_new_protected:Npn \__enumext_define_counters:Nn #1 #2
301   {
302     \cs_if_exist:cTF { c@ #2 }
303       { \msg_fatal:nnn { enumext } { counters }{ #2 } }
304       {
305         \tl_set:Nn #1 { #2 }
306         \newcounter { #2 }
307       }
308   }
```

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

enumXi
enumXii
enumXiii
enumXiv
enumXv
enumXvi
enumXvii
enumXviii

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

```
309 \__enumext_define_counters:Nn \l__enumext_counter_i_tl    { enumXi    }
310 \__enumext_define_counters:Nn \l__enumext_counter_ii_tl   { enumXii   }
311 \__enumext_define_counters:Nn \l__enumext_counter_iii_tl  { enumXiii  }
312 \__enumext_define_counters:Nn \l__enumext_counter_iv_tl   { enumXiv   }
313 \__enumext_define_counters:Nn \l__enumext_counter_v_tl    { enumXv    }
314 \__enumext_define_counters:Nn \l__enumext_counter_vi_tl   { enumXvi   }
315 \__enumext_define_counters:Nn \l__enumext_counter_vii_tl  { enumXvii  }
316 \__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.

```
317 \cs_new_protected:Npn \__enumext_register_counter_style:Nn #1 #2
318   {
319     \tl_const:cn { c__enumext_widest_ \cs_to_str:N #1 _tl } {#2}
320     \tl_gput_right:Nn \g__enumext_counter_styles_tl {#1}
321   }
322 \__enumext_register_counter_style:Nn \arabic { 0 }
323 \__enumext_register_counter_style:Nn \Alph   { M }
324 \__enumext_register_counter_style:Nn \alph   { m }
325 \__enumext_register_counter_style:Nn \Roman  { VIII }
326 \__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.

```
327 \cs_new_protected:Npn \__enumext_label_width_by_box:Nn #1 #2
328   {
329     \hbox_set:Nn \l__enumext_label_width_by_box {#2}
330     \dim_set:Nn #1 { \box_wd:N \l__enumext_label_width_by_box }
331   }
332 \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.

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

```
334    {
335      \tl_clear_new:N #1
336      \tl_put_right:Ne #1 { \tl_trim_spaces:n {#3} }
337      \tl_gset_eq:NN \g__enumext_widest_label_tl #1
338      \tl_map_inline:Nn \g__enumext_counter_styles_tl
339        {
340          \tl_replace_all:Nne #1 { ##1* } { \exp_not:N ##1 {#2} }
341          \tl_greplace_all:Nne \g__enumext_widest_label_tl { ##1* }
342            { \tl_use:c { c__enumext_widest_ \cs_to_str:N ##1 _tl } }
343        }
344      \__enumext_label_width_by_box:Nn \l__enumext_current_widest_dim
345        { \tl_use:N \g__enumext_widest_label_tl }
346      \tl_set_eq:cN { the #2 } #1
347    }
348  \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.

```
349  \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
350    {
351      \keys_define:nn { enumext / #1 }
352        {
353          font        .tl_set:c   = { l__enumext_label_font_style_#2_tl },
354          font        .value_required:n = true,
355          labelsep    .dim_set:c  = { l__enumext_labelsep_#2_dim },
356          labelsep    .initial:n  = {0.3333em},
357          labelsep    .value_required:n = true,
358          labelwidth  .dim_set:c  = { l__enumext_labelwidth_#2_dim },
359          labelwidth  .value_required:n = true,
360          wrap-label  .cs_set_protected:cp = { __enumext_wrapper_label_#2:n } ##1,
361          wrap-label  .initial:n  = {##1},
362          wrap-label  .value_required:n = true,
363          wrap-label* .code:n = {
364                              \bool_set_true:c { l__enumext_wrap_label_opt_#2_bool }
365                              \keys_set:nn { enumext / #1 } { wrap-label = {##1} }
366                            },
367          wrap-label* .value_required:n = true,
368        }
369    }
370  \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.

```
371  \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
372    {
373      \keys_define:nn { enumext / #1 }
374        {
375          align .choice:,
376          align / left   .code:n =
377                            {
378                              \tl_clear:c { l__enumext_label_fill_left_#2_tl  }
379                              \tl_set:cn  { l__enumext_label_fill_right_#2_tl } { \hfill }
380                            },
381          align / right  .code:n =
382                            {
383                              \tl_set:cn  { l__enumext_label_fill_left_#2_tl  } { \hfill }
384                              \tl_clear:c { l__enumext_label_fill_right_#2_tl }
385                            },
386          align / center .code:n =
387                            {
388                              \tl_set:cn { l__enumext_label_fill_left_#2_tl  } { \hfill }
389                              \tl_set:cn { l__enumext_label_fill_right_#2_tl } { \hfill }
390                            },
```

```
391        align .initial:n  = left,
392        align .value_required:n  = true,
393      }
394    }
395  \clist_map_inline:nn
396    {
397      {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {keyans}{v}
398    }
399    { \__enumext_tmp:nn #1 }
```

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

```
400  \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
401    {
402      \keys_define:nn { enumext / #1 }
403        {
404          align .choice:,
405          align / left    .code:n = \str_set:cn { l__enumext_align_label_#2_str } { l },
406          align / right   .code:n = \str_set:cn { l__enumext_align_label_#2_str } { r },
407          align / center  .code:n = \str_set:cn { l__enumext_align_label_#2_str } { c },
408          align .initial:n = left,
409          align .value_required:n = true,
410        }
411    }
412  \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.

```
413  \cs_new_protected:Nn \__enumext_regex_label_ref_key:
414    {
415      \tl_map_inline:Nn \c__enumext_counter_style_tl
416        {
417          \regex_replace_once:nnN { \c{##1}\* }
418            { \c{##1}\cB{\u{l__enumext_ref_aux_tl}\cE} } \l__enumext_ref_key_arg_tl
419        }
420    }
```

(*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`.

```
421  \cs_new_protected:Npn \__enumext_set_label_ref:n #1
422    {
423      \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
424      \tl_set_eq:Nc \l__enumext_ref_aux_tl { l__enumext_counter_ \__enumext_level: _tl }
425      \__enumext_regex_label_ref_key:
426      \tl_set_eq:Nc \l__enumext_ref_aux_tl { l__enumext_the_counter_ \__enumext_level: _tl }
427      \tl_put_right:ce { l__enumext_counter_style_for_ref_ \__enumext_level: _tl }
428        {
429          \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_aux_tl }
430            { \exp_not:V \l__enumext_ref_key_arg_tl }
431        }
432    }
```

(*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.

```
433  \cs_new_protected:Nn \__enumext_use_key_ref:
434    {
435      \tl_if_empty:cF { l__enumext_counter_style_for_ref_ \__enumext_level: _tl }
436        {
```

```
437          \tl_use:c { l__enumext_counter_style_for_ref_ \__enumext_level: _tl }
438        }
439    }
```

(*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.

```
440  \cs_new_protected:Npn \__enumext_set_label_ref_h:n #1
441    {
442      \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
443      \int_compare:nNnTF { \l__enumext_level_h_int } = { 1 }
444        {
445          \tl_set_eq:NN \l__enumext_ref_aux_tl \l__enumext_counter_vii_tl
446          \__enumext_regex_label_ref_key:
447          \tl_set_eq:NN \l__enumext_ref_aux_tl \l__enumext_the_counter_vii_tl
448          \tl_put_right:Ne \l__enumext_counter_style_for_ref_vii_tl
449            {
450              \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_aux_tl }
451                { \exp_not:V \l__enumext_ref_key_arg_tl }
452            }
453        }
454        {
455          \tl_set_eq:NN \l__enumext_ref_aux_tl \l__enumext_counter_viii_tl
456          \__enumext_regex_label_ref_key:
457          \tl_set_eq:NN \l__enumext_ref_aux_tl \l__enumext_the_counter_viii_tl
458          \tl_put_right:Ne \l__enumext_counter_style_for_ref_vii_tl
459            {
460              \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_aux_tl }
461                { \exp_not:V \l__enumext_ref_key_arg_tl }
462            }
463        }
464    }
```

(*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*.

```
465  \cs_new_protected:Nn \__enumext_use_key_ref_h:
466    {
467      \int_compare:nNnTF { \l__enumext_level_h_int } = { 1 }
468        {
469          \tl_if_empty:NF \l__enumext_counter_style_for_ref_vii_tl
470            {
471              \tl_use:N \l__enumext_counter_style_for_ref_vii_tl
472            }
473        }
474        {
475          \tl_if_empty:NF \l__enumext_counter_style_for_ref_viii_tl
476            {
477              \tl_use:N \l__enumext_counter_style_for_ref_viii_tl
478            }
479        }
480    }
```

(*End of definition for* __enumext_use_key_ref_h:.)

10.11.1 Define and set label key for enumext environment

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

\l__enumext_label_i_tl
\l__enumext_label_ii_tl
\l__enumext_label_iii_tl
\l__enumext_label_iv_tl

```
481  \cs_set_protected:Npn \__enumext_tmp:nnn #1 #2 #3
482    {
483      \keys_define:nn { enumext / #1 }
```

```
484          {
485            label .code:n    = {
486                                \__enumext_label_style:cvn { l__enumext_label_#2_tl }
487                                  { l__enumext_counter_#2_tl } {##1}
488                                \dim_set_eq:cN  { l__enumext_labelwidth_#2_dim }
489                                  \l__enumext_current_widest_dim
490                              },
491            label .initial:n = #3,
492            label .value_required:n = true,
493            ref   .code:n    = \__enumext_set_label_ref:n {##1},
494            ref   .value_required:n = true,
495          }
496      }
497  \__enumext_tmp:nnn { level-1 } {   i } { \arabic*.}
498  \__enumext_tmp:nnn { level-2 } {  ii } { (\alph*) }
499  \__enumext_tmp:nnn { level-3 } { iii } { \roman*. }
500  \__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.

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

```
519  \keys_define:nn { enumext / keyans }
520    {
521      label .code:n    = {
522                          \__enumext_label_style:cvn { l__enumext_label_v_tl }
523                            { l__enumext_counter_v_tl } {#1}
524                          \dim_set_eq:cN  { l__enumext_labelwidth_v_dim }
525                            \l__enumext_current_widest_dim
526                          \__enumext_label_style:cvn { l__enumext_label_vi_tl }
527                            { l__enumext_counter_vi_tl } {#1}
528                          \dim_set_eq:cN  { l__enumext_labelwidth_v_dim }
529                            \l__enumext_current_widest_dim
530                        },
531      label .initial:n = (\Alph*),
532      label .value_required:n = true,
533    }
```

(*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.

```
534 \cs_new_protected:Npn \__enumext_start_from:NNn #1 #2 #3
535     {
536       \__enumext_if_is_int:nTF { #3 }
537         {
538           \int_set:Nn #2 {#3}
539         }
540         {
541           \regex_match:nVT { \c{Alph} | \c{alph} } {#1}
542             { \int_set:Nn #2 { \int_from_alph:n {#3} } }
543           \regex_match:nVT { \c{Roman} | \c{roman} } {#1}
544             { \int_set:Nn #2  { \int_from_roman:n {#3} } }
545         }
546     }
547 \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.

```
548 \cs_new_protected:Npn \__enumext_widest_from:nNNn #1 #2 #3 #4
549     {
550       \__enumext_if_is_int:nTF {#4}
551         {
552           \setcounter{enumX#1} { #4 }
553         }
554         {
555           \regex_match:nVT { \c{Alph} | \c{alph} } {#2}
556             { \setcounter{enumX#1} { \int_from_alph:n {#4} } }
557           \regex_match:nVT { \c{Roman} | \c{roman} } {#2}
558             { \setcounter{enumX#1} { \int_from_roman:n {#4} } }
559         }
560       \__enumext_label_width_by_box:cv
561         { l__enumext_labelwidth_#1_dim } { l__enumext_label_#1_tl }
562     }
563 \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.

```
564 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
565     {
566       \keys_define:nn { enumext / #1 }
567         {
568           start  .code:n    = {
569                                 \__enumext_start_from:ccn
570                                   { l__enumext_label_#2_tl }
571                                   { l__enumext_start_#2_int } {##1}
572                               },
573           start  .initial:n = 1,
574           widest .code:n    = {
575                                 \__enumext_widest_from:nccn {#2}
576                                   { l__enumext_label_#2_tl }
577                                   { l__enumext_labelwidth_#2_dim } {##1}
578                               },
579           widest .value_required:n = true,
580           start  .value_required:n = true,
581         }
582     }
583 \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.

```
584 \cs_set_protected:Npn \__enumext_tmp:nnnnnn #1 #2 #3 #4 #5 #6
585   {
586     \keys_define:nn { enumext / #1 }
587       {
588         topsep     .skip_set:c = { l__enumext_topsep_#2_skip },
589         topsep     .initial:n  = {#3},
590         topsep     .value_required:n = true,
591         partopsep .skip_set:c = { l__enumext_partopsep_#2_skip },
592         partopsep .initial:n  = {#4},
593         partopsep .value_required:n = true,
594         parsep     .skip_set:c = { l__enumext_parsep_#2_skip },
595         parsep     .initial:n  = {#5},
596         parsep     .value_required:n = true,
597         itemsep    .skip_set:c = { l__enumext_itemsep_#2_skip },
598         itemsep    .initial:n  = {#6},
599         itemsep    .value_required:n = true,
600         noitemsep .meta:n      = { itemsep = 0pt, parsep = 0pt },
601         noitemsep .value_forbidden:n = true,
602         nosep      .meta:n      = {
603                                    itemsep = 0pt, parsep= 0pt,
604                                    topsep = 0pt, partopsep = 0pt,
605                                  },
606         nosep      .value_forbidden:n = true,
607       }
608   }
```

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

```
609 \__enumext_tmp:nnnnnn { level-1 } { i } { 8.0pt plus 2.0pt minus 4.0pt }
610   { 2.0pt plus 1.0pt minus 1.0pt } { 4.0pt plus 2.0pt minus 1.0pt }
611   { 4.0pt plus 2.0pt minus 1.0pt }
612 \__enumext_tmp:nnnnnn { level-2 } { ii } { 4.0pt plus 2.0pt minus 1.0pt }
613   { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
614   { 2.0pt plus 1.0pt minus 1.0pt }
615 \__enumext_tmp:nnnnnn { level-3 } { iii } { 2.0pt plus 1.0pt minus 1.0pt }
616   { 1.0pt minus 1.0pt }{ 0pt }{ 2.0pt plus 1.0pt minus 1.0pt }
617 \__enumext_tmp:nnnnnn { level-4 } { iv } { 2.0pt plus 1.0pt minus 1.0pt }
618   { 1.0pt minus 1.0pt }{ 0pt }{ 2.0pt plus 1.0pt minus 1.0pt }
619 \__enumext_tmp:nnnnnn { keyans  } { v }{ 4.0pt plus 2.0pt minus 1.0pt }
620   { 2.0pt plus 1.0pt minus 1.0pt }{ 2.0pt plus 1.0pt minus 1.0pt }
621   { 2.0pt plus 1.0pt minus 1.0pt }
622 \__enumext_tmp:nnnnnn { enumext* } { vii } { 8.0pt plus 2.0pt minus 4.0pt }
623   { 2.0pt plus 1.0pt minus 1.0pt } { 4.0pt plus 2.0pt minus 1.0pt }
624   { 4.0pt plus 2.0pt minus 1.0pt }
625 \__enumext_tmp:nnnnnn { keyans* } { viii } { 4.0pt plus 2.0pt minus 1.0pt }
626   { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
627   { 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.

```
628 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
629   {
630     \keys_define:nn { enumext / #1 }
631       {
632         itemindent    .dim_set:c = { l__enumext_fake_item_indent_#2_dim },
633         itemindent    .value_required:n = true,
634         rightmargin   .dim_set:c = { l__enumext_rightmargin_#2_dim },
635         rightmargin   .value_required:n = true,
636         listparindent .dim_set:c = { l__enumext_listparindent_#2_dim },
637         listparindent .value_required:n = true,
638         list-offset   .dim_set:c = { l__enumext_listoffset_#2_dim },
639         list-offset   .value_required:n = true,
640         list-indent   .code:n     =
```

```
641                              \bool_set_true:c { l__enumext_leftmargin_tmp_#2_bool }
642                              \dim_set:cn { l__enumext_leftmargin_tmp_#2_dim } {##1},
643          list-indent    .value_required:n = true,
644        }
645    }
646  \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.

```
647  \cs_set_protected:Npn \__enumext_tmp:n #1
648    {
649      \keys_define:nn { enumext / #1 } { list-indent .initial:n  = 0pt, }
650    }
651  \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.

```
652  \cs_set_protected:Nn \__enumext_fake_item:
653    {
654      \dim_compare:nNnT
655        { \dim_use:c { l__enumext_fake_item_indent_ \__enumext_level: _dim } }
656        >
657        { \c_zero_dim }
658        {
659          \tl_set:ce { l__enumext_fake_item_indent_ \__enumext_level: _tl }
660            {
661              \exp_not:N \mode_leave_vertical:
662              \exp_not:n { \skip_horizontal:n }
663                { \dim_use:c { l__enumext_fake_item_indent_ \__enumext_level: _dim } }
664              \ignorespaces
665            }
666        }
667    }
668  \cs_set_protected:Nn \__enumext_keyans_fake_item:
669    {
670      \dim_compare:nNnT
671        { \l__enumext_fake_item_indent_v_dim } > { \c_zero_dim }
672        {
673          \tl_set:Ne \l__enumext_fake_item_indent_v_tl
674            {
675              \exp_not:N \mode_leave_vertical:
676              \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_v_dim
677            }
678        }
679    }
680  \cs_set_protected:Nn \__enumext_fake_item_vii:
681    {
682      \dim_compare:nNnT
683        { \l__enumext_fake_item_indent_vii_dim } > { \c_zero_dim }
684        {
685          \tl_set:Ne \l__enumext_fake_item_indent_vii_tl
686            {
687              \exp_not:N \mode_leave_vertical:
688              \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_vii_dim
689            }
690        }
691    }
692  \cs_set_protected:Nn \__enumext_fake_item_viii:
693    {
694      \dim_compare:nNnT
695        { \l__enumext_fake_item_indent_viii_dim } > { \c_zero_dim }
696        {
697          \tl_set:Ne \l__enumext_fake_item_indent_viii_tl
698            {
699              \exp_not:N \mode_leave_vertical:
700              \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_viii_dim
```

```
701            }
702         }
703      }
```

(*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.

```
704  \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
705    {
706      \keys_define:nn { enumext / #1 }
707        {
708          show-length .bool_set:c = { l__enumext_show_length_#2_bool },
709          show-length .initial:n  = false,
710        }
711    }
712  \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.

```
713  \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
714    {
715      \keys_define:nn { enumext / #1 }
716        {
717          before  .tl_set:c   = { l__enumext_before_no_starred_key_#2_tl },
718          before  .value_required:n = true,
719          before* .tl_set:c   = { l__enumext_before_starred_key_#2_tl },
720          before* .value_required:n = true,
721          after   .tl_set:c   = { l__enumext_after_stop_list_#2_tl },
722          after   .value_required:n = true,
723          first   .tl_set:c   = { l__enumext_after_list_args_#2_tl },
724          first   .value_required:n = true,
725        }
726    }
727  \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.

```
728  \cs_new_protected:Nn \__enumext_before_args_exec:
729    {
730      \tl_use:c { l__enumext_before_starred_key_ \__enumext_level: _tl }
731    }
```

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*⟩.

```
732  \cs_new_protected:Nn \__enumext_before_keys_exec:
733    {
734      \tl_use:c { l__enumext_before_no_starred_key_ \__enumext_level: _tl }
735    }
```

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

```
736  \cs_new_protected:Nn \__enumext_after_stop_list:
737    {
738      \tl_use:c { l__enumext_after_stop_list_ \__enumext_level: _tl }
739    }
```

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`.

```
740 \cs_new_protected:Nn \__enumext_after_args_exec:
741   {
742     \tl_use:c { l__enumext_after_list_args_ \__enumext_level: _tl }
743   }
```

(*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.

```
744 \cs_new_protected:Nn \__enumext_before_args_exec_v:
745   {
746     \tl_use:N \l__enumext_before_starred_key_v_tl
747   }
```

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*⟩.

```
748 \cs_new_protected:Nn \__enumext_before_keys_exec_v:
749   {
750     \tl_use:N \l__enumext_before_no_starred_key_v_tl
751   }
```

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

```
752 \cs_new_protected:Nn \__enumext_after_stop_list_v:
753   {
754     \tl_use:N \l__enumext_after_stop_list_v_tl
755   }
```

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`.

```
756 \cs_new_protected:Nn \__enumext_after_args_exec_v:
757   {
758     \tl_use:N \l__enumext_after_list_args_v_tl
759   }
```

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

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

`\__enumext_before_args_exec_vii:`
`\__enumext_before_keys_exec_vii`
`\__enumext_after_stop_list_vii:`
`\__enumext_after_args_exec_vii:`

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

```
760 \cs_new_protected:Nn \__enumext_before_args_exec_vii:
761   {
762     \tl_use:N \l__enumext_before_starred_key_vii_tl
763   }
764 \cs_new_protected:Nn \__enumext_before_args_exec_viii:
765   {
766     \tl_use:N \l__enumext_before_starred_key_viii_tl
767   }
```

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*⟩.

```
768 \cs_new_protected:Nn \__enumext_before_keys_exec_vii:
769   {
770     \tl_use:N \l__enumext_before_no_starred_key_vii_tl
771   }
772 \cs_new_protected:Nn \__enumext_before_keys_exec_viii:
773   {
774     \tl_use:N \l__enumext_before_no_starred_key_viii_tl
775   }
```

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

```
776  \cs_new_protected:Nn \__enumext_after_stop_list_vii:
777    {
778      \tl_use:N \l__enumext_after_stop_list_vii_tl
779    }
780  \cs_new_protected:Nn \__enumext_after_stop_list_viii:
781    {
782      \tl_use:N \l__enumext_after_stop_list_viii_tl
783    }
```

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.

```
784  \cs_new_protected:Nn \__enumext_after_args_exec_vii:
785    {
786      \tl_use:N \l__enumext_after_list_args_vii_tl
787    }
788  \cs_new_protected:Nn \__enumext_after_args_exec_viii:
789    {
790      \tl_use:N \l__enumext_after_list_args_viii_tl
791    }
```

(*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.

```
792  \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
793    {
794      \keys_define:nn { enumext / #1 }
795        {
796          mini-env    .dim_set:c   = { l__enumext_minipage_right_#2_dim },
797          mini-env    .value_required:n = true,
798          mini-sep    .dim_set:c   = { l__enumext_minipage_hsep_#2_dim },
799          mini-sep    .initial:n   = 0.3333em,
800          mini-sep    .value_required:n = true,
801          columns-sep .dim_set:c   = { l__enumext_columns_sep_#2_dim },
802          columns-sep .value_required:n = true,
803          columns     .int_set:c   = { l__enumext_columns_#2_int },
804          columns     .initial:n   = 1,
805          columns     .value_required:n = true,
806        }
807    }
808  \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.

```
809  \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
810    {
811      \keys_define:nn { enumext / #1 }
812        {
813          columns    .initial:n = 2,
814          miniright  .tl_gset:c = { g__enumext_miniright_code_#2_tl },
815          miniright  .value_required:n = true,
816          miniright* .code:n    = {
817                                    \bool_gset_true:c { g__enumext_minipage_center_#2_bool }
818                                    \keys_set:nn { enumext / #1 } { miniright = {##1} }
819                                  },
820          miniright* .value_required:n = true,
821        }
822    }
823  \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.
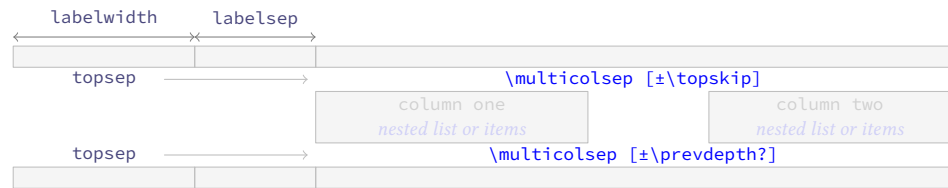
Figure 7: Representation of the vertical space in `multicols` for a nested level.

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*.

```
824  \cs_new_protected:Nn \__enumext_multi_set_vskip:
825    {
826      \skip_set:cn { l__enumext_multicols_above_ \__enumext_level: _skip }
827        {
828          \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
829        }
830      \skip_set:cn { l__enumext_multicols_below_ \__enumext_level: _skip }
831        {
832          \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
833        }
834      \__enumext_add_pre_parsep:
835    }
```

(*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*.

```
836  \cs_new_protected:Nn \__enumext_add_pre_parsep:
837    {
838      \int_case:nn { \l__enumext_level_int }
839        {
840          { 2 }{
841              \skip_if_eq:nnF { \l__enumext_parsep_i_skip } { \c_zero_skip }
842                {
843                  \skip_add:Nn \l__enumext_multicols_above_ii_skip { \l__enumext_parsep_i_skip }
844                }
845            }
846          { 3 }{
847              \skip_if_eq:nnF { \l__enumext_parsep_ii_skip } { \c_zero_skip }
848                {
849                  \skip_add:Nn \l__enumext_multicols_above_iii_skip { \l__enumext_parsep_ii_skip
850                }
851            }
852          { 4 }{
853              \skip_if_eq:nnF { \l__enumext_parsep_iii_skip } { \c_zero_skip }
854                {
855                  \skip_add:Nn \l__enumext_multicols_above_iv_skip { \l__enumext_parsep_iii_skip
856                }
857            }
```

```
858            }
859        }
```

(*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*⟩.

```
860  \cs_new_protected:Nn \__enumext_multi_addvspace:
861    {
862      \__enumext_multi_set_vskip:
863      \mode_if_vertical:T
864        {
865          \skip_add:cn { l__enumext_multicols_above_ \__enumext_level: _skip }
866            {
867              \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
868            }
869          \skip_add:cn { l__enumext_multicols_below_ \__enumext_level: _skip }
870            {
871              \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
872            }
873        }
874      \par\nopagebreak
875      \addvspace{ \skip_use:c { l__enumext_multicols_above_ \__enumext_level: _skip } }
876    }
```

(*End of definition for* __enumext_multi_addvspace:.)

10.18.2   Adjustment of vertical spaces for multicols in keyans

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

```
877  \cs_new_protected:Nn \__enumext_keyans_multi_set_vskip:
878    {
879      \skip_set:Nn \l__enumext_multicols_above_v_skip
880        {
881          \l__enumext_topsep_v_skip
882        }
883      \skip_set:Nn \l__enumext_multicols_below_v_skip
884        {
885          \l__enumext_topsep_v_skip
886        }
887    }
888  \cs_new_protected:Nn \__enumext_keyans_multi_addvspace:
889    {
890      \__enumext_keyans_multi_set_vskip:
891      \mode_if_vertical:T
892        {
893          \skip_add:Nn \l__enumext_multicols_above_v_skip
894            {
895              \skip_use:N \l__enumext_partopsep_v_skip
896            }
897          \skip_add:Nn \l__enumext_multicols_below_v_skip
898            {
899              \skip_use:N \l__enumext_partopsep_v_skip
900            }
901        }
902      \par\nopagebreak
903      \addvspace{ \l__enumext_multicols_above_v_skip }
904    }
```

(*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.

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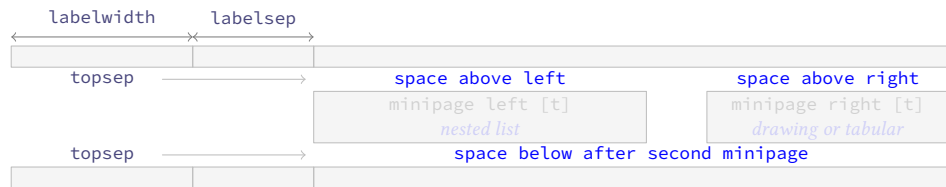
Figure 8: Representation of the `minipage` spacing adjustment for a nested level.

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.

```
905  \DeclareDocumentEnvironment{__enumext_mini_env*}{ m }
906    {
907      \__enumext_minipage:w [ t ] { #1 }
908        \legacy_if_gset_false:n { @minipage }
909        \vspace { 0pt }
910    }
911    { \__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`.

```
912  \cs_new_protected:Nn \__enumext_mini_set_vskip:
913    {
914      \int_compare:nNnTF
915        { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } > { 1 }
916        {
```

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.

```
917          \skip_if_eq:nnTF
918            { \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip } } { \c_zero_skip }
919            {
920              \skip_set:Nn \l__enumext_minipage_left_skip
921                {
922                  -0.150\box_dp:N \strutbox
923                }
924              \skip_set:Nn \l__enumext_minipage_right_skip
925                {
926                  0.695\box_dp:N \strutbox
927                }
928              \skip_set:Nn \l__enumext_minipage_after_skip
929                {
930                  \box_dp:N \strutbox
931                }
932              \__enumext_zero_parsep:
933            }
```

```
934                    {
935                        \skip_set:Nn \l__enumext_minipage_left_skip
936                            {
937                                \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
938                            }
939                        \skip_set:Nn \l__enumext_minipage_right_skip
940                            {
941                                0.695\box_dp:N \strutbox
942                            }
943                        \skip_set:Nn \l__enumext_minipage_after_skip
944                            {
945                                1.85\box_dp:N \strutbox
946                                + \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
947                            }
948                    }
949                }
950                {
```

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.

```
951                    \skip_if_eq:nnTF
952                        { \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip } } { \c_zero_skip }
953                        {
954                            \skip_set:Nn \l__enumext_minipage_left_skip
955                                {
956                                    0.5\box_dp:N \strutbox
957                                    - \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
958                                }
959                            \skip_set:Nn \l__enumext_minipage_right_skip
960                                {
961                                    \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
962                                }
963                            \skip_set:Nn \l__enumext_minipage_after_skip
964                                {
965                                    1.6\box_dp:N \strutbox
966                                }
967                        }
968                        {
969                            \skip_set:Nn \l__enumext_minipage_left_skip
970                                {
971                                    0.5875\box_dp:N \strutbox
972                                    - \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
973                                }
974                            \skip_set:Nn \l__enumext_minipage_right_skip
975                                {
976                                    + \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
977                                    + \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
978                                }
979                            \skip_set:Nn \l__enumext_minipage_after_skip
980                                {
981                                    0.325\box_dp:N \strutbox
982                                    + \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
983                                }
984                        }
985                }
986            }
```

(*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.

```
987    \cs_new_protected:Nn \__enumext_zero_parsep:
988        {
989            \int_case:nn { \l__enumext_level_int }
990                {
991                    { 2 }{
992                        \skip_if_eq:nnF { \l__enumext_parsep_i_skip } { \c_zero_skip }
993                            {
994                                \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
```

```
995                    }
996                 }
997           { 3 }{
998                 \skip_if_eq:nnF { \l__enumext_parsep_ii_skip } { \c_zero_skip }
999                    {
1000                      \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
1001                    }
1002                 }
1003          { 4 }{
1004                 \skip_if_eq:nnF { \l__enumext_parsep_iii_skip } { \c_zero_skip }
1005                    {
1006                      \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
1007                    }
1008                 }
1009          }
1010       }
```

(*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*.

```
1011  \cs_new_protected:Nn \__enumext_mini_addvspace:
1012    {
1013      \__enumext_mini_set_vskip:
1014      \mode_if_vertical:T
1015        {
1016          \skip_add:Nn \l__enumext_minipage_left_skip
1017            {
1018              \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
1019            }
1020          \skip_add:Nn \l__enumext_minipage_after_skip
1021            {
1022              \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
1023            }
1024        }
1025      \par\nopagebreak
1026      \addvspace { \l__enumext_minipage_left_skip }
1027    }
```

(*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.

```
1028  \cs_new_protected:Nn \__enumext_keyans_mini_set_vskip:
1029    {
1030      \skip_zero_new:N \l__enumext_minipage_after_skip
1031      \skip_zero_new:N \l__enumext_minipage_left_skip
1032      \skip_zero_new:N \l__enumext_minipage_right_skip
1033      \int_compare:nNnTF { \l__enumext_columns_v_int } > { 1 }
1034        {
1035          \skip_if_eq:nnTF { \l__enumext_topsep_v_skip } { \c_zero_skip }
1036            {
1037              \skip_set:Nn \l__enumext_minipage_left_skip { -0.25\box_dp:N \strutbox }
1038              \skip_set:Nn \l__enumext_minipage_right_skip { 0.705\box_dp:N \strutbox }
1039              \skip_set:Nn \l__enumext_minipage_after_skip { \box_dp:N \strutbox }
1040              \skip_if_eq:nnF { \l__enumext_parsep_i_skip } { \c_zero_skip }
1041                {
1042                  \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
1043                }
1044            }
1045            {
1046              \skip_set:Nn \l__enumext_minipage_left_skip
1047                {
1048                  \skip_use:N \l__enumext_topsep_v_skip
1049                }
1050              \skip_set:Nn \l__enumext_minipage_right_skip
```

```
1051              {
1052                  0.705\box_dp:N \strutbox
1053              }
1054          \skip_set:Nn \l__enumext_minipage_after_skip
1055              {
1056                  1.85\box_dp:N \strutbox + \l__enumext_topsep_v_skip
1057              }
1058          }
1059      }
1060      {
1061          \skip_if_eq:nnTF { \l__enumext_topsep_v_skip } { \c_zero_skip }
1062              {
1063              \skip_set:Nn \l__enumext_minipage_left_skip
1064                  {
1065                      0.5\box_dp:N \strutbox
1066                      + \l__enumext_partopsep_v_skip
1067                  }
1068              \skip_set:Nn \l__enumext_minipage_right_skip
1069                  {
1070                      \l__enumext_partopsep_v_skip
1071                  }
1072              \skip_set:Nn \l__enumext_minipage_after_skip { 1.6\box_dp:N \strutbox }
1073              }
1074              {
1075              \skip_set:Nn \l__enumext_minipage_left_skip
1076                  {
1077                      0.5875\box_dp:N \strutbox - \l__enumext_partopsep_v_skip
1078                  }
1079              \skip_set:Nn \l__enumext_minipage_right_skip
1080                  {
1081                      \l__enumext_topsep_v_skip + \l__enumext_partopsep_v_skip
1082                  }
1083              \skip_set:Nn \l__enumext_minipage_after_skip
1084                  {
1085                      0.325\box_dp:N \strutbox + \l__enumext_topsep_v_skip
1086                  }
1087              }
1088      }
1089  }
```

(*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.

```
1090  \cs_new_protected:Nn \__enumext_keyans_mini_addvspace:
1091      {
1092          \__enumext_keyans_mini_set_vskip:
1093          \mode_if_vertical:T
1094              {
1095              \skip_add:Nn \l__enumext_minipage_left_skip
1096                  {
1097                      \l__enumext_partopsep_v_skip
1098                  }
1099              \skip_add:Nn \l__enumext_minipage_after_skip
1100                  {
1101                      \l__enumext_partopsep_v_skip
1102                  }
1103              }
1104          \par\nopagebreak
1105          \addvspace { \l__enumext_minipage_left_skip }
1106      }
```

(*End of definition for* __enumext_keyans_mini_addvspace:.)

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

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

```
1107  \cs_new_protected:Nn \__enumext_mini_set_vskip_vii:
1108    {
1109      \skip_zero_new:N \l__enumext_minipage_left_skip
1110      \skip_gzero_new:N \g__enumext_minipage_right_skip
1111      \skip_gzero_new:N \g__enumext_minipage_after_skip
1112      \skip_if_eq:nnTF { \l__enumext_topsep_vii_skip } { \c_zero_skip }
1113        {
1114          \skip_set:Nn \l__enumext_minipage_left_skip { 0.5\box_dp:N \strutbox }
1115          \skip_gset:Nn \g__enumext_minipage_right_skip { 0.325\box_dp:N \strutbox }
1116        }
1117        {
1118          \skip_set:Nn \l__enumext_minipage_left_skip { 0.5875\box_dp:N \strutbox }
1119          \skip_gset:Nn \g__enumext_minipage_right_skip
1120            {
1121              \l__enumext_topsep_vii_skip
1122            }
1123          \skip_gset:Nn \g__enumext_minipage_after_skip
1124            {
1125              0.325\box_dp:N \strutbox + \l__enumext_topsep_vii_skip
1126            }
1127        }
1128    }
1129  \cs_new_protected:Nn \__enumext_mini_set_vskip_viii:
1130    {
1131      \skip_zero_new:N \l__enumext_minipage_after_skip
1132      \skip_zero_new:N \l__enumext_minipage_left_skip
1133      \skip_zero_new:N \l__enumext_minipage_right_skip
1134      \skip_if_eq:nnTF { \l__enumext_topsep_viii_skip } { \c_zero_skip }
1135        {
1136          \skip_set:Nn \l__enumext_minipage_left_skip
1137            {
1138              0.5\box_dp:N \strutbox
1139            }
1140          \skip_set:Nn \l__enumext_minipage_right_skip
1141            {
1142              \l__enumext_partopsep_viii_skip
1143            }
1144          \skip_set:Nn \l__enumext_minipage_after_skip
1145            {
1146              1.6\box_dp:N \strutbox
1147            }
1148        }
1149        {
1150          \skip_set:Nn \l__enumext_minipage_left_skip
1151            {
1152              0.5875\box_dp:N \strutbox
1153            }
1154          \skip_set:Nn \l__enumext_minipage_right_skip
1155            {
1156              \l__enumext_topsep_viii_skip
1157            }
1158          \skip_set:Nn \l__enumext_minipage_after_skip
1159            {
1160              0.325\box_dp:N \strutbox + \l__enumext_topsep_viii_skip
1161            }
1162        }
1163    }
```

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

__enumext_mini_addvspace_vii:
__enumext_mini_addvspace_viii:

The functions __enumext_mini_addvspace_vii: and __enumext_mini_addvspace_viii: will apply the vertical space *"only above"* the __enumext_mini_env* environment on the *left side* when the miniright key is active in the enumext* and keyans* environments.

Here we will NOT take into account whether TeX is in ⟨*horizontal mode*⟩ or ⟨*vertical mode*⟩, since \partopsep is equal to 0pt in both environments.

```
1164  \cs_new_protected:Nn \__enumext_mini_addvspace_vii:
1165    {
1166      \__enumext_mini_set_vskip_vii:
1167      \par\nopagebreak
1168      \addvspace { \l__enumext_minipage_left_skip }
1169    }
```

```
1170  \cs_new_protected:Nn \__enumext_mini_addvspace_viii:
1171    {
1172      \__enumext_mini_set_vskip_viii:
1173      \par\nopagebreak
1174      \addvspace { \l__enumext_minipage_left_skip }
1175    }
```

(*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.

```
1176  \NewDocumentCommand \miniright { s }
1177    {
1178      \int_compare:nNnT { \l__enumext_keyans_pic_level_int } = { 1 }
1179        {
1180          \msg_error:nnn { enumext } { wrong-miniright-place }
1181        }
1182      \int_compare:nNnT { \l__enumext_level_int } = { 0 }
1183        {
1184          \msg_error:nnn { enumext } { wrong-miniright-place }
1185        }
1186      \int_compare:nNnTF { \l__enumext_keyans_level_int } = { 1 }
1187        {
1188          \__enumext_keyans_mini_right_cmd:n {#1}
1189        }
1190        { \__enumext_mini_right_cmd:n {#1} }
1191    }
```

(*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.

```
1192  \cs_new_protected:Npn \__enumext_mini_right_cmd:n #1
1193    {
1194      \dim_compare:nNnTF
1195        { \dim_use:c { l__enumext_minipage_right_ \__enumext_level: _dim } } > { \c_zero_dim }
1196        {
1197          \__enumext_multicols_stop:
1198          \end{__enumext_mini_env*}
1199          \hfill
1200          \begin{__enumext_mini_env*}
1201            { \dim_use:c { l__enumext_minipage_right_ \__enumext_level: _dim } }
1202            \par\addvspace { \l__enumext_minipage_right_skip }
1203            \bool_if:nF {#1}
1204              {
1205                \centering
1206              }
1207            \int_gzero:N \g__enumext_minipage_stat_int
1208        }
1209        { \msg_error:nnn { enumext } { wrong-miniright-use } }
1210    }
```

(*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.

```
1211  \cs_new_protected:Npn \__enumext_keyans_mini_right_cmd:n #1
1212    {
1213      \dim_compare:nNnTF { \l__enumext_minipage_right_v_dim } > { \c_zero_dim }
1214        {
1215          \__enumext_keyans_multicols_stop:
1216          \end{__enumext_mini_env*}
1217          \hfill
1218          \begin{__enumext_mini_env*}{ \l__enumext_minipage_right_v_dim }
1219            \par\addvspace { \l__enumext_minipage_right_skip }
1220            \bool_if:nF {#1}
1221              {
1222                \centering
1223              }
1224            \int_gzero:N \g__enumext_minipage_stat_int
1225        }
1226        { \msg_error:nnn { enumext } { wrong-miniright-use } }
1227    }
```

(*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.

```
1228  \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1229    {
1230      \keys_define:nn { enumext / #1 }
1231        {
1232          above  .skip_set:c = { l__enumext_vspace_above_#2_skip },
1233          above  .value_required:n = true,
1234          above* .code:n     = \bool_set_true:c { l__enumext_vspace_a_star_#2_bool }
1235                                 \keys_set:nn { enumext / #1 } { above = {##1} },
1236          above* .value_required:n = true,
1237          below  .skip_set:c = { l__enumext_vspace_below_#2_skip },
1238          below  .value_required:n = true,
1239          below* .code:n     = \bool_set_true:c { l__enumext_vspace_b_star_#2_bool }
1240                                 \keys_set:nn { enumext / #1 } { below = {##1} },
1241          below* .value_required:n = true,
1242        }
1243    }
1244  \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.

```
1245  \cs_new_protected:Nn \__enumext_vspace_above:
1246    {
1247      \skip_if_eq:nnF
1248        { \skip_use:c { l__enumext_vspace_above_ \__enumext_level: _skip } } { \c_zero_skip }
1249        {
1250          \bool_if:cTF { l__enumext_vspace_a_star_ \__enumext_level: _bool }
1251            {
1252              \vspace*{ \skip_use:c { l__enumext_vspace_above_ \__enumext_level: _skip } }
1253            }
1254            {
1255              \vspace { \skip_use:c { l__enumext_vspace_above_ \__enumext_level: _skip } }
1256            }
1257        }
1258    }
```

(*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.

```
1259 \cs_new_protected:Nn \__enumext_vspace_below:
1260   {
1261     \skip_if_eq:nnF
1262       { \skip_use:c { l__enumext_vspace_below_ \__enumext_level: _skip } } { \c_zero_skip }
1263       {
1264         \bool_if:cTF { l__enumext_vspace_b_star_ \__enumext_level: _bool }
1265           {
1266             \vspace*{ \skip_use:c { l__enumext_vspace_below_ \__enumext_level: _skip } }
1267           }
1268           {
1269             \vspace { \skip_use:c { l__enumext_vspace_below_ \__enumext_level: _skip } }
1270           }
1271       }
1272   }
```

(*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.

```
1273 \cs_new_protected:Nn \__enumext_vspace_above_v:
1274   {
1275     \skip_if_eq:nnF { \l__enumext_vspace_above_v_skip } { \c_zero_skip }
1276       {
1277         \bool_if:NTF \l__enumext_vspace_a_star_v_bool
1278           {
1279             \vspace*{ \l__enumext_vspace_above_v_skip }
1280           }
1281           { \vspace { \l__enumext_vspace_above_v_skip } }
1282       }
1283   }
```

(*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.

```
1284 \cs_new_protected:Nn \__enumext_vspace_below_v:
1285   {
1286     \skip_if_eq:nnF { \l__enumext_vspace_below_v_skip } { \c_zero_skip }
1287       {
1288         \bool_if:NTF \l__enumext_vspace_b_star_v_bool
1289           {
1290             \vspace*{ \l__enumext_vspace_below_v_skip }
1291           }
1292           { \vspace { \l__enumext_vspace_below_v_skip } }
1293       }
1294   }
```

(*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.

```
1295 \cs_new_protected:Nn \__enumext_vspace_above_vii:
1296   {
1297     \skip_if_eq:nnF { \l__enumext_vspace_above_vii_skip } { \c_zero_skip }
1298       {
1299         \bool_if:NTF \l__enumext_vspace_a_star_vii_bool
1300           {
1301             \vspace*{ \l__enumext_vspace_above_vii_skip }
1302           }
1303           { \vspace { \l__enumext_vspace_above_vii_skip } }
1304       }
1305   }
1306 \cs_new_protected:Nn \__enumext_vspace_above_viii:
1307   {
1308     \skip_if_eq:nnF { \l__enumext_vspace_above_viii_skip } { \c_zero_skip }
1309       {
```

```
1310          \bool_if:NTF \l__enumext_vspace_a_star_viii_bool
1311            {
1312              \vspace*{ \l__enumext_vspace_above_viii_skip }
1313            }
1314            { \vspace { \l__enumext_vspace_above_viii_skip } }
1315        }
1316    }
```

(*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.

```
1317 \cs_new_protected:Nn \__enumext_vspace_below_vii:
1318    {
1319      \skip_if_eq:nnF { \l__enumext_vspace_below_vii_skip } { \c_zero_skip }
1320        {
1321          \bool_if:NTF \l__enumext_vspace_b_star_vii_bool
1322            {
1323              \vspace*{ \l__enumext_vspace_below_vii_skip }
1324            }
1325            { \vspace { \l__enumext_vspace_below_vii_skip } }
1326        }
1327    }
1328 \cs_new_protected:Nn \__enumext_vspace_below_viii:
1329    {
1330      \skip_if_eq:nnF { \l__enumext_vspace_below_viii_skip } { \c_zero_skip }
1331        {
1332          \bool_if:NTF \l__enumext_vspace_b_star_viii_bool
1333            {
1334              \vspace*{ \l__enumext_vspace_below_viii_skip }
1335            }
1336            { \vspace { \l__enumext_vspace_below_viii_skip } }
1337        }
1338    }
```

(*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*.

```
1339 \keys_define:nn { enumext / level-1 }
1340    {
1341      save-ans .code:n = \__enumext_storing_set:n {#1},
1342      save-ans .value_required:n = true,
1343      resume   .code:n = \__enumext_resume_counter:,
1344      resume   .value_forbidden:n = true,
1345      resume*  .code:n = \__enumext_resume_counter_star:,
1346      resume*  .value_forbidden:n = true,
1347    }

1348 \keys_define:nn { enumext / enumext* }
1349    {
1350      save-ans .code:n = \__enumext_storing_set:n {#1},
1351      save-ans .value_required:n = true,
1352      resume   .code:n = \__enumext_resume_counter_vii:,
1353      resume   .value_forbidden:n = true,
1354    }
```

(*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.

```
1355 \cs_new_protected:Npn \__enumext_storing_set:n #1
1356   {
1357     \tl_set:Ne \l__enumext_store_name_tl {#1}
1358     \prop_if_exist:cF { g__enumext_ \l__enumext_store_name_tl _prop }
1359       {
1360         \prop_new:c { g__enumext_ \l__enumext_store_name_tl _prop }
1361       }
1362     \seq_if_exist:cF { g__enumext_ \l__enumext_store_name_tl _seq }
1363       {
1364         \seq_new:c { g__enumext_ \l__enumext_store_name_tl _seq }
1365       }
1366     \bool_set_true:N \l__enumext_store_active_bool
1367     \bool_set_true:N \l__enumext_store_ans_bool
1368     \int_if_exist:cF { g__enumext_resume_#1_int }
1369       {
1370         \int_new:c { g__enumext_resume_#1_int }
1371       }
1372   }
```

(*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.

```
1373 \cs_new_protected:Nn \__enumext_resume_counter:
1374   {
1375     \bool_if:NT \l__enumext_store_active_bool
1376       {
1377         \int_gset:Nn \g__enumext_resume_int
1378           {
1379             \int_use:c { g__enumext_resume_ \l__enumext_store_name_tl _int }
1380           }
1381       }
1382     \bool_set_true:N \l__enumext_resume_bool
1383   }
1384 \cs_new_protected:Nn \__enumext_resume_counter_vii:
1385   {
1386     \bool_if:NT \l__enumext_store_active_bool
1387       {
1388         \int_gset:Nn \g__enumext_resume_int
1389           {
1390             \int_use:c { g__enumext_resume_ \l__enumext_store_name_tl _int }
1391           }
1392       }
1393     \bool_set_true:N \l__enumext_resume_vii_bool
1394   }
```

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

10.22 The check answer mechanism

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

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

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

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

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

The integer variable associated to the sum of each `\item` and `\item*` in the environment `\g__enumext_-count_item_number_int` must match the integer variable `\g__enumext_count_item_anskey_int` associated to the execution of the command `\anskey`. We analyze the cases:

a) If the list only has one level the number of `\item` + `\item*` = `\anskey`
b) If the list has *nested levels*, for each level of nesting we need to decrementing by one (for the `\item` or `\item*` that opens the nest) so that the account remains the same.

With `keyans`, `keyans*` and `keyanspic` it is enough to increase in one the integer of `\anskey`. The integers created must be global if they are not lost in the interior levels of nesting and to execute the test we will use a *"hook"* function after closing the first level of the environment.

10.22.1 Setting check-ans key

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

```
1395 \cs_set_protected:Npn \__enumext_tmp:n #1
1396   {
1397     \keys_define:nn { enumext / #1 }
1398       {
1399         check-ans .bool_set:N = \l__enumext_check_ans_bool,
1400         check-ans .initial:n  = false,
1401         no-store  .code:n = {
1402                             \bool_set_false:N \l__enumext_store_ans_bool
1403                             \bool_set_false:N \l__enumext_check_ans_bool
1404                           },
1405         no-store  .value_forbidden:n = true,
1406       }
1407   }
1408 \clist_map_inline:nn
1409   {
1410     level-1, level-2, level-3, level-4, enumext*
1411   }
1412   { \__enumext_tmp:n {#1} }
```

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

10.22.2 Set-up check answer mechanism

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

```
1413 \cs_new_protected:Nn \__enumext_check_ans_set:
1414   {
1415     \int_case:nn { \l__enumext_level_int }
1416       {
1417         { 1 }{
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_if:NT \g__enumext_standar_bool
1442               {
1443                 \int_gdecr:N \g__enumext_count_item_number_int
1444                 \typeout{ENUMEXT ~ STARRED ~ NEEEEEEEEEEEESTED}
```

```
1445                          }
1446                      }
1447                  }
1448          }
```

(*End of definition for* __enumext_check_ans_set:.)

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

```
1449  \cs_new_protected:Nn \__enumext_check_ans_exec:
1450    {
1451      \bool_if:NT \l__enumext_check_ans_bool
1452        {
1453          \__enumext_check_ans_set:
1454        }
1455    }
```

(*End of definition for* __enumext_check_ans_exec:.)

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

```
1456  \cs_new_protected:Nn \__enumext_check_ans_show:
1457    {
1458      \int_compare:nNnTF
1459        { \g__enumext_count_item_number_int } = { \g__enumext_count_item_anskey_int }
1460        {
1461          \msg_term:nnV { enumext } { items-same-answer } \g__enumext_store_name_tl
1462        }
1463        {
1464          \msg_warning:nnV { enumext } { item-different-answer } \g__enumext_store_name_tl
1465        }
1466      \int_gzero:N \g__enumext_count_item_number_int
1467      \int_gzero:N \g__enumext_count_item_anskey_int
1468    }
```

(*End of definition for* __enumext_check_ans_show:.)

10.23   Keys and functions associated with storage

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

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

```
1469  \cs_set_protected:Npn \__enumext_tmp:n #1
1470    {
1471      \keys_define:nn { enumext / #1 }
1472        {
1473          wrap-ans   .cs_set_protected:Np = \__enumext_anskey_wrapper:n ##1,
1474          wrap-ans   .initial:n = \fbox{##1},
1475          wrap-ans   .value_required:n = true,
1476          wrap-opt   .cs_set_protected:Np = \__enumext_keyans_wrapper_opt:n ##1,
1477          wrap-opt   .initial:n = [{##1}],
1478          wrap-opt   .value_required:n = true,
1479          save-sep   .tl_set:N  = \l__enumext_store_keyans_item_opt_sep_tl,
1480          save-sep   .initial:n = {, ~ },
1481          save-sep   .value_required:n = true,
1482          mark-ans   .tl_set:N  = \l__enumext_mark_answer_sym_tl,
1483          mark-ans   .initial:n = \textasteriskcentered,
1484          mark-ans   .value_required:n = true,
1485          mark-pos   .choice:,
1486          mark-pos  / left   .code:n = \str_set:Nn \l__enumext_mark_position_str { l },
1487          mark-pos  / right  .code:n = \str_set:Nn \l__enumext_mark_position_str { r },
1488          mark-pos   .initial:n = right,
1489          mark-pos   .value_required:n = true,
1490          show-ans   .bool_set:N = \l__enumext_show_answer_bool,
1491          show-ans   .initial:n  = false,
1492          show-ans   .value_required:n = true,
1493          show-pos   .bool_set:N = \l__enumext_show_position_bool,
1494          show-pos   .initial:n  = false,
1495          show-pos   .value_required:n = true,
```

```
1496        mark-ref   .tl_set:N   = \l__enumext_mark_ref_sym_tl,
1497        mark-ref   .initial:n  = \textasteriskcentered,
1498        mark-ref   .value_required:n = true,
1499        save-ref   .bool_set:N = \l__enumext_store_ref_key_bool,
1500        save-ref   .initial:n  = false,
1501        save-ref   .value_required:n = true,
1502      }
1503    }
1504 \clist_map_inline:nn { level-1, enumext* } { \__enumext_tmp:n {#1} }
```

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

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

```
1505 \cs_set_protected:Npn \__enumext_tmp:n #1
1506   {
1507     \keys_define:nn { enumext / #1 }
1508       {
1509         mark-pos .choice:,
1510         mark-pos / left  .code:n = \str_set:Nn \l__enumext_mark_position_str { l },
1511         mark-pos / right .code:n = \str_set:Nn \l__enumext_mark_position_str { r },
1512         mark-pos .initial:n = right,
1513         mark-pos .value_required:n  = true,
1514         show-ans .bool_set:N = \l__enumext_show_answer_bool,
1515         show-ans .initial:n  = false,
1516         show-ans .value_required:n = true,
1517         show-pos .bool_set:N = \l__enumext_show_position_bool,
1518         show-pos .initial:n  = false,
1519         show-pos .value_required:n = true,
1520       }
1521   }
1522 \clist_map_inline:nn { keyans, keyans* } { \__enumext_tmp:n {#1} }
```

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

columns*   For the `enumext` and `enumext*` environments we will only add the keys `columns*` and `columns-sep*`.
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.

```
1523 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1524   {
1525     \keys_define:nn { enumext / #1 }
1526       {
1527         columns*     .code:n = \bool_set_true:c { l__enumext_store_columns_#2_bool }
1528                               \int_set:cn { l__enumext_store_columns_#2_int } {##1}
1529                               \tl_put_right:ce { l__enumext_store_opt_#2_tl }
1530                                 {
1531                                   columns = \exp_not:v { l__enumext_store_columns_#2_int },
1532                                 },
1533         columns*     .value_required:n  = true,
1534         columns-sep* .code:n = \bool_set_true:c { l__enumext_store_columns_sep_#2_bool }
1535                               \dim_set:cn { l__enumext_store_columns_sep_#2_dim } {##1}
1536                               \tl_put_right:ce { l__enumext_store_opt_#2_tl }
1537                                 {
1538                                   columns-sep = \exp_not:v { l__enumext_store_columns_sep_#2_di
1539                                 },
1540         columns-sep* .value_required:n  = true,
1541       }
1542   }
1543 \clist_map_inline:nn
1544   {
1545     {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {enumext*}{vii}
1546   }
1547   { \__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.

```
1548 \cs_generate_variant:Nn \prop_gput_if_not_in:Nnn { cen }
1549 \cs_new_protected:Npn \__enumext_store_addto_prop:n #1
1550   {
1551     \prop_gput_if_not_in:cen { g__enumext_ \l__enumext_store_name_tl _prop }
1552       {
1553         \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } + 1 }
1554       }
1555       { #1 }
1556   }
1557 \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.

```
1558 \cs_new_protected:Npn \__enumext_store_addto_seq:n #1
1559   {
1560     \seq_gput_right:cn { g__enumext_ \l__enumext_store_name_tl _seq } { #1 }
1561   }
1562 \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`.

```
1563 \cs_new_protected:Nn \__enumext_store_level_open:
1564   {
1565     \bool_if:NT \l__enumext_store_ans_bool
1566       {
1567         \tl_if_empty:cTF { l__enumext_store_opt_ \__enumext_level: _tl }
1568           {
1569             \__enumext_store_addto_seq:n
1570               {
1571                 \item \begin{enumext}
1572               }
1573           }
1574           {
1575             \tl_put_left:cn { l__enumext_store_opt_ \__enumext_level: _tl }
1576               {
1577                 \item \begin{enumext} [
1578               }
1579             \tl_put_right:cn { l__enumext_store_opt_ \__enumext_level: _tl }
1580               {
1581                 ]
1582               }
1583             \__enumext_store_addto_seq:v { l__enumext_store_opt_ \__enumext_level: _tl }
1584           }
1585       }
1586   }
1587 \cs_new_protected:Nn \__enumext_store_level_close:
1588   {
1589     \bool_if:NT \l__enumext_store_ans_bool
1590       {
```

```
1591              \__enumext_store_addto_seq:n { \end{enumext} }
1592          }
1593    }
```

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

__enumext_store_level_open_vii: When nesting the enumext* environment in enumext starting right after \item (without material between
__enumext_store_level_close_vii: 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.

```
1594 \cs_new_protected:Nn \__enumext_store_level_open_vii:
1595    {
1596      \bool_if:NT \l__enumext_store_ans_bool
1597        {
1598          \tl_if_empty:NTF \l__enumext_store_opt_vii_tl
1599            {
1600              \__enumext_store_addto_seq:n
1601                {
1602                  \item \mode_leave_vertical:
1603                    \vspace { -\skip_eval:n { \baselineskip + \parsep } }
1604                    \begin{enumext*}[before={\setlength{\topsep}{0pt}},]
1605                }
1606            }
1607            {
1608              \tl_put_left:Nn \l__enumext_store_opt_vii_tl
1609                {
1610                  \item \mode_leave_vertical:
1611                    \vspace { -\skip_eval:n { \baselineskip + \parsep } }
1612                    \begin{enumext*}[before={\setlength{\topsep}{0pt}},
1613                }
1614              \tl_put_right:Nn \l__enumext_store_opt_vii_tl
1615                {
1616                  ]
1617                }
1618              \__enumext_store_addto_seq:V \l__enumext_store_opt_vii_tl
1619            }
1620        }
1621    }
1622 \cs_new_protected:Nn \__enumext_store_level_close_vii:
1623    {
1624      \bool_if:NT \l__enumext_store_ans_bool
1625        {
1626          \__enumext_store_addto_seq:n { \end{enumext*} }
1627        }
1628    }
```

(*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 The function __enumext_print_keyans_box:NN print a box in the left margin with \l__enumext_-
__enumext_print_keyans_box:cc 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

```
1629 \cs_new_protected:Nn \__enumext_print_keyans_box:NN
1630    {
1631      \mode_leave_vertical:
1632      \skip_horizontal:n { -\dim_use:N #2 }
1633      \makebox[0pt][ r ]
1634        {
1635          \makebox[ \dim_use:N #1 ][ \l__enumext_mark_position_str ]
1636            {
1637              \tl_use:N \l__enumext_mark_answer_sym_tl
1638            }
1639        }
1640      \skip_horizontal:n { \dim_use:N #2 }
1641    }
1642 \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`.

```
1643 \keys_define:nn { enumext / anskey }
1644   {
1645     item-sym* .tl_set:N   = \l__enumext_store_item_symbol_tl,
1646     item-sym* .value_required:n = true,
1647     item-pos* .dim_set:N  = \l__enumext_store_item_symbol_sep_dim,
1648     item-pos* .value_required:n = true,
1649     store-brk .bool_set:N = \l__enumext_store_columns_break_bool,
1650     store-brk .default:n  = true,
1651     store-brk .value_forbidden:n = true,
1652   }
```

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*⟩.

```
1653 \NewDocumentCommand \anskey { d() s o +m }
1654   {
1655     \bool_if:NF \l__enumext_store_active_bool
1656       {
1657         \msg_error:nnnn { enumext } { anskey-wrong-place }{ anskey }{ enumext }
1658       }
1659     \int_compare:nNnT { \l__enumext_keyans_level_int } = { 1 }
1660       {
1661         \msg_error:nnnn { enumext } { command-wrong-place }{ anskey }{ keyans }
1662       }
1663     \int_compare:nNnT { \l__enumext_keyans_pic_level_int } = { 1 }
1664       {
1665         \msg_error:nnnn { enumext } { command-wrong-place }{ anskey }{ keyanspic }
1666       }
1667     \group_begin:
1668       \bool_if:NT \l__enumext_store_ans_bool
1669         {
1670           \bool_if:NT \l__enumext_check_ans_bool
1671             {
1672               \int_gincr:N \g__enumext_count_item_anskey_int
1673             }
1674           \__enumext_store_anskey_code:nnnn {#1} {#2} {#3} {#4}
1675         }
1676     \group_end:
1677   }
```

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

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

```
1678 \cs_new_protected:Npn \__enumext_store_anskey_code:nnnn #1 #2 #3 #4
1679   {
1680     \__enumext_store_addto_prop:n {#4}
1681     \bool_if:NT \l__enumext_store_ref_key_bool
1682       {
1683         \__enumext_store_internal_ref:
1684       }
1685     \__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`.

```
1686        \tl_clear:N \l__enumext_store_anskey_arg_tl
1687        \tl_if_novalue:nF {#3}
1688          {
1689            \keys_set:nn { enumext / anskey } {#3}
1690          }
1691        \bool_lazy_and:nnT
1692          { \bool_if_p:N \l__enumext_store_columns_break_bool }
1693          { \bool_not_p:n { \l__enumext_starred_bool } }
1694          {
1695            \tl_put_left:Nn \l__enumext_store_anskey_arg_tl { \columnbreak }
1696          }
1697        \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).

```
1698        \tl_if_novalue:nF {#1}
1699          {
1700            \int_set:Nn \l__enumext_store_columns_join_int {#1}
1701            \bool_if:NT \l__enumext_starred_bool
1702              {
1703                \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
1704                  {
1705                    ( \exp_not:V \l__enumext_store_columns_join_int )
1706                  }
1707              }
1708          }
```

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.

```
1709        \bool_if:nTF {#2}
1710          {
1711            \tl_put_right:Nn \l__enumext_store_anskey_arg_tl { * }
1712            \tl_if_empty:NF \l__enumext_store_item_symbol_tl
1713              {
1714                \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
1715                  {
1716                    [ \exp_not:V \l__enumext_store_item_symbol_tl ]
1717                  }
1718              }
1719            \dim_compare:nT
1720              {
1721                \l__enumext_store_item_symbol_sep_dim != \c_zero_dim
1722              }
1723              {
1724                \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
1725                  {
1726                    [ \exp_not:V \l__enumext_store_item_symbol_sep_dim ]
1727                  }
1728              }
1729            \tl_put_right:Nn \l__enumext_store_anskey_arg_tl {#4}
1730          }
1731          {
1732            \tl_put_right:Nn \l__enumext_store_anskey_arg_tl {#4}
1733          }
```

Finally we check if the save-ref key is active along with the hyperref package load, if both conditions are met, it will create the \hyperlink and then store in ⟨*sequence*⟩.

```
1734        \bool_lazy_and:nnT
1735          { \bool_if_p:N \l__enumext_store_ref_key_bool }
1736          { \bool_if_p:N \l__enumext_hyperref_bool }
1737          {
1738            \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
1739              {
1740                \hfill \exp_not:N \hyperlink { \exp_not:V \l__enumext_newlabel_arg_one_tl }
1741                  { \exp_not:V \l__enumext_mark_ref_sym_tl }
1742              }
1743          }
1744        \__enumext_store_addto_seq:V \l__enumext_store_anskey_arg_tl
1745      }
```

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

__enumext_store_internal_ref:

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

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

```
1746 \cs_new_protected:Nn \__enumext_store_internal_ref:
1747   {
1748     \cs_set_protected:Npn \__enumext_tmp:n ##1
1749       {
1750         \tl_set_eq:cc { l__enumext_label_copy_##1_tl } { l__enumext_label_##1_tl }
1751         \tl_reverse:c { l__enumext_label_copy_##1_tl }
1752         \tl_remove_once:cn { l__enumext_label_copy_##1_tl } { . }
1753         \tl_reverse:c { l__enumext_label_copy_##1_tl }
1754       }
1755     \clist_map_inline:nn { i, ii, iii, iv, vii } { \__enumext_tmp:n {##1} }
1756     \cs_set:Npn \__enumext_tmp:n ##1
1757       { . \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.

```
1758     \bool_lazy_all:nT
1759       {
1760         { \bool_if_p:N \g__enumext_starred_bool }
1761         { \int_compare_p:nNn { \l__enumext_level_int } = { \c_zero_int } }
1762       }
1763       {
1764         \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
1765           { \tl_use:N \l__enumext_label_copy_vii_tl }
1766       }
1767     \bool_lazy_all:nT
1768       {
1769         { \bool_if_p:N \l__enumext_standar_bool }
1770         { \bool_if_p:N \g__enumext_starred_bool }
1771         { \int_compare_p:nNn { \l__enumext_level_int } > { \c_zero_int } }
1772       }
1773       {
1774         \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
1775           {
1776             \tl_use:N \l__enumext_label_copy_vii_tl
1777             \int_step_function:nnN { 1 } { \l__enumext_level_int } \__enumext_tmp:n
1778           }
1779       }
```

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.

```
1780     \bool_lazy_all:nT
1781       {
1782         { \bool_if_p:N \l__enumext_standar_bool }
1783         { \int_compare_p:nNn { \l__enumext_level_int } > { \c_zero_int } }
1784         { \int_compare_p:nNn { \l__enumext_level_h_int } = { \c_zero_int } }
1785         { \bool_not_p:n { \l__enumext_starred_bool } }
1786       }
1787       {
1788         \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
1789           {
1790             \tl_use:N \l__enumext_label_copy_i_tl
1791             \int_step_function:nnN { 2 } { \l__enumext_level_int } \__enumext_tmp:n
1792           }
1793       }
1794     \cs_set:Npn \__enumext_tmp:n ##1
1795       { \tl_use:c { l__enumext_label_copy_ \int_to_roman:n {##1} _tl } }
1796     \bool_lazy_all:nT
1797       {
1798         { \bool_if_p:N \l__enumext_standar_bool }
1799         { \int_compare_p:nNn { \l__enumext_level_int } > { \c_zero_int } }
1800         { \bool_not_p:n { \g__enumext_starred_bool } }
1801         { \int_compare_p:nNn { \l__enumext_level_h_int } > { \c_zero_int } }
1802       }
1803       {
1804         \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
1805           {
```

```
1806                  \int_step_function:nnN { 1 } { \l__enumext_level_int } \__enumext_tmp:n
1807                  . \tl_use:N \l__enumext_label_copy_vii_tl
1808              }
1809          }
```

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

```
1810      \tl_put_right:Ne \l__enumext_newlabel_arg_one_tl
1811          {
1812              \l__enumext_store_name_tl \c_colon_str
1813              \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } }
1814          }
```

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.

```
1815      \tl_put_right:Ne \l__enumext_store_write_aux_file_tl
1816          {
1817              \__enumext_newlabel:nn
1818                  { \exp_not:V \l__enumext_newlabel_arg_one_tl }
1819                  { \l__enumext_newlabel_arg_two_tl }
1820          }
1821      \l__enumext_store_write_aux_file_tl
1822  }
```

(*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.

```
1823  \cs_new_protected:Npn \__enumext_store_anskey_show_wrap:n #1
1824      {
1825          \par
1826          \bool_if:NT \l__enumext_starred_bool
1827              {
1828                  \cs_set:Nn \__enumext_level: { vii }
1829              }
1830          \__enumext_print_keyans_box:cc
1831              { l__enumext_labelwidth_ \__enumext_level: _dim }
1832              { l__enumext_labelsep_ \__enumext_level: _dim }
1833          \__enumext_anskey_wrapper:n { #1 }
1834      }
```

(*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.

```
1835  \cs_new_protected:Npn \__enumext_store_anskey_show_left:n #1
1836      {
1837          \bool_if:NT \l__enumext_show_answer_bool
1838              {
1839                  \__enumext_store_anskey_show_wrap:n { #1 }
1840              }
1841          \bool_if:NT \l__enumext_show_position_bool
1842              {
1843                  \tl_set:Ne \l__enumext_mark_answer_sym_tl
1844                      {
1845                          \group_begin:
1846                          \exp_not:N \normalfont
1847                          \exp_not:N \footnotesize [ \int_eval:n
1848                              {
1849                                  \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
1850                              }
1851                          ]
1852                          \group_end:
1853                      }
1854              \__enumext_store_anskey_show_wrap:n { #1 }
1855          }
1856  }
```

(*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`.

```
1857 \cs_new_protected:Npn \__enumext_keyans_addto_prop:n #1
1858   {
1859     \tl_clear:N \l__enumext_store_keyans_label_tl
1860     \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
1861       {
1862         \tl_put_right:Ne \l__enumext_store_keyans_label_tl { \l__enumext_label_vi_tl }
1863       }
1864       {
1865         \tl_put_right:Ne \l__enumext_store_keyans_label_tl { \l__enumext_label_v_tl }
1866       }
1867     \tl_if_novalue:nF { #1 }
1868       {
1869         % Set save-sep
1870         \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_tl
1871           {
1872             \tl_put_right:Ne \l__enumext_store_keyans_label_tl { \l__enumext_store_keyans_item_opt
1873           }
1874         \tl_put_right:Ne \l__enumext_store_keyans_label_tl { #1 }
1875       }
1876     \__enumext_store_addto_prop:V \l__enumext_store_keyans_label_tl
1877   }
```

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

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

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

__enumext_keyans_store_ref:
__enumext_keyans_store_ref_aux_i:
__enumext_keyans_store_ref_aux_ii:

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

```
1878 \cs_new_protected:Nn \__enumext_keyans_store_ref:
1879   {
1880     \bool_if:NT \l__enumext_store_ref_key_bool
1881       {
1882         \cs_set_protected:Npn \__enumext_tmp:n ##1
1883           {
1884             \tl_set_eq:cc { l__enumext_label_copy_##1_tl } { l__enumext_label_##1_tl }
1885             \tl_reverse:c { l__enumext_label_copy_##1_tl }
1886             \tl_remove_once:cn { l__enumext_label_copy_##1_tl } { . }
1887             \tl_reverse:c { l__enumext_label_copy_##1_tl }
1888           }
1889         \clist_map_inline:nn { i, v, vi, vii, viii } { \__enumext_tmp:n {##1} }
1890         \__enumext_keyans_store_ref_aux_i:
1891       }
1892   }
```

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`.

```
1893 \cs_new_protected:Nn \__enumext_keyans_store_ref_aux_i:
1894   {
1895     \bool_if:NT \g__enumext_starred_bool
1896       {
1897         \tl_set_eq:NN \l__enumext_label_copy_i_tl \l__enumext_label_copy_vii_tl
1898       }
1899     \int_compare:nNnT { \l__enumext_keyans_pic_level_int } = { 1 }
1900       {
1901         \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
```

```
1902                    { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_vi_tl }
1903            }
1904        \int_compare:nNnT { \l__enumext_keyans_level_int } = { 1 }
1905            {
1906                \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
1907                    { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_v_tl }
1908            }
1909        \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
1910            {
1911                \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
1912                    { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_viii_tl }
1913            }
1914        \tl_put_right:Ne \l__enumext_newlabel_arg_one_tl
1915            {
1916                \l__enumext_store_name_tl \c_colon_str
1917                \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } }
1918            }
1919        \__enumext_keyans_store_ref_aux_ii:
1920    }
```

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.

```
1921 \cs_new_protected:Nn \__enumext_keyans_store_ref_aux_ii:
1922    {
1923        \tl_put_right:Ne \l__enumext_store_write_aux_file_tl
1924            {
1925                \__enumext_newlabel:nn
1926                    { \exp_not:V \l__enumext_newlabel_arg_one_tl }
1927                    { \l__enumext_newlabel_arg_two_tl }
1928            }
1929        \l__enumext_store_write_aux_file_tl
1930    }
```

(*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.

```
1931 \cs_new_protected:Npn \__enumext_keyans_addto_seq:n #1
1932    {
1933        \tl_clear:N \l__enumext_store_keyans_label_tl
1934        \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
1935            {
1936                \tl_put_right:Ne \l__enumext_store_keyans_label_tl { \item \l__enumext_label_vi_tl }
1937            }
1938            {
1939                \tl_put_right:Ne \l__enumext_store_keyans_label_tl { \item \l__enumext_label_v_tl }
1940            }
1941        \tl_if_novalue:nF { #1 }
1942            {
1943                \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_tl
1944                    {
1945                        \tl_put_right:Ne \l__enumext_store_keyans_label_tl { \l__enumext_store_keyans_item_op
1946                    }
1947                \tl_put_right:Ne \l__enumext_store_keyans_label_tl { #1 }
1948            }
1949        \__enumext_keyans_addto_seq_link:
1950    }
```

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

```
1951 \cs_new_protected:Nn \__enumext_keyans_addto_seq_link:
1952    {
```

```
1953       \bool_lazy_and:nnT
1954         { \bool_if_p:N \l__enumext_store_ref_key_bool }
1955         { \bool_if_p:N \l__enumext_hyperref_bool }
1956         {
1957           \tl_put_right:Ne \l__enumext_store_keyans_label_tl
1958             {
1959               \hfill \exp_not:N \hyperlink
1960                 {
1961                   \exp_not:V \l__enumext_newlabel_arg_one_tl
1962                 }
1963                 { \exp_not:V \l__enumext_mark_ref_sym_tl }
1964             }
1965         }
1966       \__enumext_store_addto_seq:V \l__enumext_store_keyans_label_tl
1967       \tl_gset:NV \g__enumext_check_ans_item_tl \l__enumext_store_keyans_label_tl
1968       \bool_if:NT \l__enumext_check_ans_bool
1969         {
1970           \int_gincr:N \g__enumext_count_item_anskey_int
1971         }
1972     }
```

(*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.

```
1973 \cs_new_protected:Npn \__enumext_keyans_check_ans:nn #1 #2
1974   {
1975     \tl_if_empty:NTF \g__enumext_check_ans_item_tl
1976       {
1977         \msg_warning:nnnn { enumext } { missing-starred }{ #1 }{ #2 }
1978       }
1979       { \tl_gclear:N \g__enumext_check_ans_item_tl }
1980   }
```

(*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`
`\__enumext_keyans_show_ans:`
`\__enumext_keyans_show_pos:`
`\__enumext_keyans_show_item_opt:`

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

```
1981 \cs_new_protected:Npn \__enumext_keyans_show_left:n #1
1982   {
1983     \tl_if_novalue:nF { #1 }
1984       {
1985         \tl_set:Ne \l__enumext_keyans_item_opt_tl { #1 }
1986       }
1987     \bool_if:NT \l__enumext_show_answer_bool
1988       {
1989         \__enumext_keyans_show_ans:
1990       }
1991     \bool_if:NT \l__enumext_show_position_bool
1992       {
1993         \__enumext_keyans_show_pos:
1994       }
1995   }
1996 \cs_new_protected:Nn \__enumext_keyans_show_item_opt:
1997   {
1998     \tl_if_empty:NF \l__enumext_keyans_item_opt_tl
1999       {
2000         \bool_lazy_or:nnT
2001           { \bool_if_p:N \l__enumext_show_answer_bool }
2002           { \bool_if_p:N \l__enumext_show_position_bool }
2003           {
2004             \__enumext_keyans_wrapper_opt:n { \l__enumext_keyans_item_opt_tl } \c_space_tl
2005           }
2006       }
```

```
2007        }
2008  \cs_new_protected:Nn \__enumext_keyans_show_ans:
2009    {
2010       \tl_put_left:Nn \l__enumext_label_v_tl
2011         {
2012            \__enumext_print_keyans_box:NN \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
2013         }
2014    }
2015  \cs_new_protected:Nn \__enumext_keyans_show_pos:
2016    {
2017       \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
2018         {
2019            \tl_set:Ne \l__enumext_mark_answer_sym_tl
2020              {
2021                 \group_begin:
2022                 \exp_not:N \normalfont
2023                 \exp_not:N \footnotesize [ \int_eval:n
2024                   {
2025                      \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
2026                   }
2027                   ]
2028                 \group_end:
2029              }
2030         }
2031         {
2032            \tl_set:Ne \l__enumext_mark_answer_sym_tl
2033              {
2034                 \group_begin:
2035                 \exp_not:N \normalfont
2036                 \exp_not:N \footnotesize [ \int_eval:n
2037                   {
2038                      \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } + 1
2039                   }
2040                   ]
2041                 \group_end:
2042              }
2043         }
2044       \tl_put_left:Nn \l__enumext_label_v_tl
2045         {
2046            \__enumext_print_keyans_box:NN
2047              \l__enumext_labelwidth_i_dim
2048              \l__enumext_labelsep_i_dim
2049         }
2050    }
```

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

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* Define and set `item-sym*` and `item-pos*` keys for enumext and enumext*.
item-pos*
```
2051  \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
2052    {
2053       \keys_define:nn { enumext / #1 }
2054         {
2055            item-sym* .tl_set:c  = { l__enumext_item_symbol_#2_tl },
2056            item-sym* .value_required:n = true,
2057            item-sym* .initial:n  = {$\star$},
2058            item-pos* .dim_set:c  = { l__enumext_item_symbol_sep_#2_dim },
2059            item-pos* .value_required:n = true,
2060         }
2061    }
2062  \clist_map_inline:nn
2063    {
2064       {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {enumext*}{vii}
2065    }
2066    { \__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.

```
2067  \cs_new_protected:Nn \__enumext_footnotetext:nn
2068    {
2069      \footnotetext[#1]{#2}
2070    }
2071  \cs_new_protected:Nn \__enumext_renew_footnote:
2072    {
2073      \seq_gclear:N \g__enumext_footnote_arg_seq
2074      \seq_gclear:N \g__enumext_footnote_int_seq
2075      \RenewDocumentCommand \footnote { o +m }
2076        {
2077          \tl_if_novalue:nTF {##1}
2078            {
2079              \stepcounter{footnote}
2080              \int_gset_eq:Nc \g__enumext_footnote_int { c@footnote }
2081            }
2082            {
2083              \int_gset:Nn \g__enumext_footnote_int { ##1 }
2084            }
2085          \footnotemark [ \g__enumext_footnote_int ]
2086          \seq_gput_right:Nn \g__enumext_footnote_arg_seq { ##2 }
2087          \seq_gput_right:NV \g__enumext_footnote_int_seq \g__enumext_footnote_int
2088        }
2089    }
2090  \cs_new_protected:Nn \__enumext_print_footnote:
2091    {
2092      \seq_if_empty:NF \g__enumext_footnote_int_seq
2093        {
2094          \seq_map_pairwise_function:NNN
2095            \g__enumext_footnote_int_seq
2096            \g__enumext_footnote_arg_seq
2097            \__enumext_footnotetext:nn
2098        }
2099    }
```

(*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).

```
2100  \cs_new_protected:Npn \__enumext_default_item:n #1
2101    {
2102      \tl_if_novalue:nTF {#1}
2103        {
2104          \bool_if:NT \l__enumext_check_ans_bool
2105            {
2106              \int_gincr:N \g__enumext_count_item_number_int
2107            }
2108          \bool_set_true:c { l__enumext_wrap_label_ \__enumext_level: _bool }
2109          \__enumext_item_std:w \tl_use:c { l__enumext_fake_item_indent_ \__enumext_level: _tl }
2110        }
2111        {
2112          \bool_set_eq:cc
2113            { l__enumext_wrap_label_ \__enumext_level: _bool }
```

```
2114          { l__enumext_wrap_label_opt_ \__enumext_level: _bool }
2115            \__enumext_item_std:w [#1] \tl_use:c { l__enumext_fake_item_indent_ \__enumext_level: _tl
2116          }
2117      }
```

(*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).

```
2118  \cs_new_protected:Npn \__enumext_starred_item:nn #1 #2
2119    {
2120      \tl_if_novalue:nF {#1}
2121        {
2122          \tl_set:cn { l__enumext_item_symbol_ \__enumext_level: _tl } {#1}
2123        }
2124      \tl_gset_eq:Nc \g__enumext_item_symbol_tl { l__enumext_item_symbol_ \__enumext_level: _tl }
2125      \tl_if_novalue:nTF {#2}
2126        {
2127          \dim_set_eq:cc
2128            { l__enumext_item_symbol_sep_ \__enumext_level: _dim }
2129            { l__enumext_labelsep_ \__enumext_level: _dim }
2130        }
2131        {
2132          \dim_set:cn { l__enumext_item_symbol_sep_ \__enumext_level: _dim } {#2}
2133        }
2134      \bool_if:NT \l__enumext_check_ans_bool
2135        {
2136          \int_gincr:N \g__enumext_count_item_number_int
2137        }
2138      \bool_set_true:c { l__enumext_wrap_label_ \__enumext_level: _bool }
2139      \__enumext_item_std:w \tl_use:c { l__enumext_fake_item_indent_ \__enumext_level: _tl }
2140    }
```

(*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).

```
2141  \cs_new_protected:Nn \__enumext_redefine_item:
2142    {
2143      \RenewDocumentCommand \item { s o o }
2144        {
2145          \bool_if:nTF {##1}
2146            {
2147              \__enumext_starred_item:nn {##2} {##3}
2148            }
2149            { \__enumext_default_item:n {##2} }
2150        }
2151    }
```

(*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.

```
2152 \cs_new_protected:Npn \__enumext_keyans_default_item:n #1
2153   {
2154     \tl_if_novalue:nTF { #1 }
2155       {
2156         \bool_set_true:N \l__enumext_wrap_label_v_bool
2157         \__enumext_item_std:w \tl_use:N \l__enumext_fake_item_indent_v_tl
2158       }
2159       {
2160         \bool_set_eq:NN \l__enumext_wrap_label_v_bool \l__enumext_wrap_label_opt_v_bool
2161         \__enumext_item_std:w [#1] \tl_use:N \l__enumext_fake_item_indent_v_tl
2162       }
2163   }
```

(*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⟩.

```
2164 \cs_new_protected:Npn \__enumext_keyans_starred_item:n #1
2165   {
2166     \tl_set_eq:NN \l__enumext_keyans_tmpa_tl \l__enumext_label_v_tl
2167     \__enumext_keyans_show_left:n { #1 }
2168     \bool_set_true:N \l__enumext_wrap_label_v_bool
2169     \__enumext_item_std:w \tl_use:N \l__enumext_fake_item_indent_v_tl \__enumext_keyans_show_item_
```

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

```
2170     \tl_set_eq:NN \l__enumext_label_v_tl \l__enumext_keyans_tmpa_tl
2171     \__enumext_keyans_addto_prop:n { #1 }
2172     \__enumext_keyans_store_ref:
2173     \__enumext_keyans_addto_seq:n { #1 }
2174   }
```

(*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).

```
2175 \cs_new_protected:Nn \__enumext_keyans_redefine_item:
2176   {
2177     \RenewDocumentCommand \item { s o }
2178       {
2179         \bool_if:nTF {##1}
2180           {
2181             \peek_remove_spaces:n
2182               {
2183                 \__enumext_keyans_starred_item:n {##2}
2184               }
2185           }
2186           {
2187             \__enumext_keyans_default_item:n {##2}
2188           }
2189       }
2190   }
```

(*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.

```
2191  \cs_new_protected:Nn \__enumext_item_starred:
2192    {
2193      \tl_if_empty:cF { l__enumext_item_symbol_ \__enumext_level: _tl }
2194        {
2195          \mode_leave_vertical:
2196          \skip_horizontal:n { -\dim_use:c { l__enumext_item_symbol_sep_ \__enumext_level: _dim } }
2197          \makebox[ 0pt ][ r ]{ \g__enumext_item_symbol_tl }
2198          \skip_horizontal:n { \dim_use:c { l__enumext_item_symbol_sep_ \__enumext_level: _dim } }
2199        }
2200    }
```

(*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).

```
2201  \cs_new_protected:Nn \__enumext_make_label:
2202    {
2203      \RenewDocumentCommand \makelabel { m }
2204        {
2205          \tl_use:c { l__enumext_label_fill_left_ \__enumext_level: _tl }
2206          \tl_use:c { l__enumext_label_font_style_ \__enumext_level: _tl }
2207          \bool_if:cTF { l__enumext_wrap_label_ \__enumext_level: _bool }
2208            {
2209              \__enumext_item_starred:
2210              \use:c { __enumext_wrapper_label_ \__enumext_level: :n } { ##1 }
2211            }
2212            { ##1 }
2213          \tl_use:c { l__enumext_label_fill_right_ \__enumext_level: _tl }
2214          \tl_gclear:N \g__enumext_item_symbol_tl
2215        }
2216    }
```

(*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).

```
2217  \cs_new_protected:Nn \__enumext_keyans_make_label:
2218    {
2219      \RenewDocumentCommand \makelabel { m }
2220        {
2221          \tl_use:N \l__enumext_label_fill_left_v_tl
2222          \tl_use:N \l__enumext_label_font_style_v_tl
2223          \bool_if:NTF \l__enumext_wrap_label_v_bool
2224            {
2225              \__enumext_wrapper_label_v:n { ##1 }
2226            }
2227            { ##1 }
2228          \tl_use:N \l__enumext_label_fill_right_v_tl
2229        }
2230    }
```

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

__enumext_calc_hspace:NNNNNNN
__enumext_calc_hspace:ccccccc

The function __enumext_calc_hspace:NNNNNNN takes seven arguments to be able to determine horizontal spaces for all list environment:
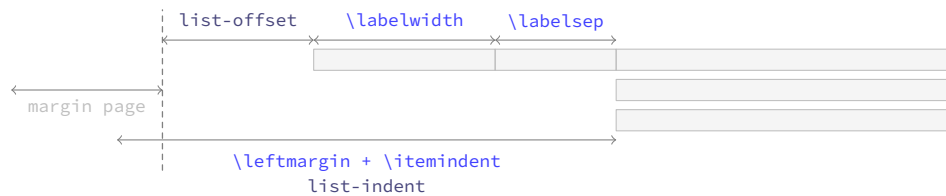
Figure 9: Representation of standard horizontal lengths in `list` environment.
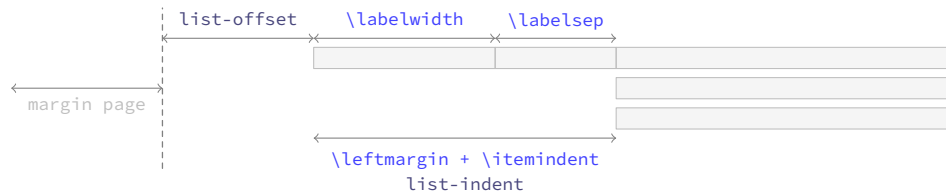
Figure 10: Representation of horizontal lengths concept in list in enumext.

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

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

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

```
2231 \cs_new_protected:Npn \__enumext_calc_hspace:NNNNNNN #1 #2 #3 #4 #5 #6 #7
2232   {
2233     \dim_compare:nNnT { #1 } < { \c_zero_dim }
2234       {
2235         \msg_warning:nnnV { enumext } { width-non-positive }{ labelwidth }{ #1 }
2236         \dim_set:Nn #1 { \dim_abs:n { #1 } }
2237       }
2238     \dim_compare:nNnT { #2 } < { \c_zero_dim }
2239       {
2240         \msg_warning:nnnV { enumext } { width-negative }{ labelsep }{ #2 }
2241         \dim_set:Nn #2 { \dim_abs:n { #2 } }
2242       }
```

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

```
2243       \bool_if:nF #7 { \dim_set:Nn #4 { #1 + #2} }
```

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

```
2244       \dim_compare:nNnTF { #4 } < { \c_zero_dim }
2245         {
2246           \dim_set:Nn #6 { #1 + #2 - #4}
2247           \dim_set:Nn #5 { #1 + #2 + #3 - #6 }
2248         }
2249         {
2250           \dim_compare:nNnT { #4 } = { #1 + #2 }
2251             { \dim_set:Nn #6 { \c_zero_dim } }
2252           \dim_compare:nNnT { #4 } < { #1 + #2 }
2253             { \dim_set:Nn #6 { #1 + #2 - #4} }
2254           \dim_compare:nNnT { #4 } > { #1 + #2 }
2255             {
2256               \dim_set:Nn #6 { -#1 - #2 + #4}
2257               \dim_set:Nn #6 { #6*-1}
2258             }
```
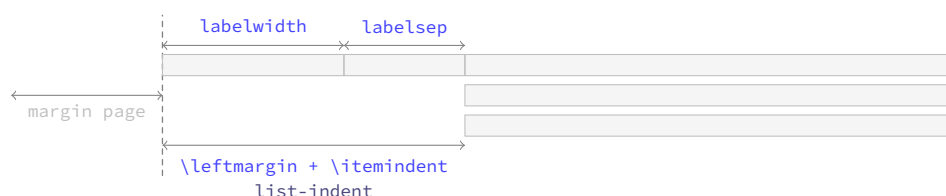

Figure 11: Default horizontal lengths in enumext.

```
2259          \dim_set:Nn #5 { #1 + #2 + #3 - #6 }
2260        }
2261    }
2262 \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.

```
2263 \cs_set_protected:Npn \__enumext_tmp:n #1
2264   {
2265     \cs_new_protected:cpn { __enumext_list_arg_two_#1: }
2266       {
2267         \__enumext_calc_hspace:ccccccc
2268           { l__enumext_labelwidth_#1_dim } { l__enumext_labelsep_#1_dim }
2269           { l__enumext_listoffset_#1_dim } { l__enumext_leftmargin_tmp_#1_dim }
2270           { l__enumext_leftmargin_#1_dim } { l__enumext_itemindent_#1_dim }
2271           { l__enumext_leftmargin_tmp_#1_bool }
2272         \clist_map_inline:nn
2273           { labelsep, labelwidth, itemindent, leftmargin, rightmargin, listparindent }
2274           { \dim_set_eq:cc {####1} { l__enumext_####1_#1_dim } }
2275         \clist_map_inline:nn { topsep, parsep, partopsep, itemsep }
2276           { \skip_set_eq:cc {####1} { l__enumext_####1_#1_skip } }
2277         \usecounter { enumX#1 }
2278         \bool_lazy_and:nnTF
2279           { \str_if_eq_p:nn {#1} { i } }
2280           { \bool_if_p:N \l__enumext_resume_bool }
2281           { \setcounter { enumXi  } { \int_eval:n { \g__enumext_resume_int } } }
2282           {
2283             \setcounter { enumX#1 }
2284               { \int_eval:n { \int_use:c { l__enumext_start_#1_int } - 1 } }
2285           }
2286         \str_if_eq:nnTF {#1} { v }
2287           {
2288             \__enumext_keyans_redefine_item:
2289             \__enumext_keyans_make_label:
2290             \__enumext_keyans_fake_item:
2291             \bool_if:cT { l__enumext_show_length_#1_bool }
2292               {
2293                 \msg_term:nnnn { enumext } { list-lengths-not-nested } { v } { keyans }
2294               }
2295           }
2296           {
2297             \__enumext_redefine_item:
2298             \__enumext_make_label:
2299             \__enumext_use_key_ref:
2300             \__enumext_fake_item:
2301             \bool_if:cT { l__enumext_show_length_#1_bool }
2302               {
2303                 \msg_term:nnne { enumext } { list-lengths } {#1} { \int_use:N \l__enumext_level_i
2304               }
2305           }
2306       }
2307   }
2308 \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.

```
2309  \cs_set_protected:Npn \__enumext_tmp:n #1
2310    {
2311      \cs_new_protected:cpn { __enumext_list_arg_two_#1: }
2312        {
2313          \__enumext_calc_hspace:ccccccc
2314            { l__enumext_labelwidth_#1_dim } { l__enumext_labelsep_#1_dim }
2315            { l__enumext_listoffset_#1_dim } { l__enumext_leftmargin_tmp_#1_dim }
2316            { l__enumext_leftmargin_#1_dim } { l__enumext_itemindent_#1_dim }
2317            { l__enumext_leftmargin_tmp_#1_bool }
2318          \clist_map_inline:nn
2319            { labelsep, labelwidth, itemindent, leftmargin, rightmargin, listparindent }
2320            { \dim_set_eq:cc {####1} { l__enumext_####1_#1_dim } }
2321          \clist_map_inline:nn { topsep, parsep, partopsep, itemsep }
2322            { \skip_set_eq:cc {####1} { l__enumext_####1_#1_skip } }
2323          \skip_set_eq:Nc \parsep  { l__enumext_itemsep_#1_skip }
2324          \skip_zero:N \partopsep
2325          \usecounter { enumX#1 }
2326          \bool_lazy_and:nnTF
2327            { \str_if_eq_p:nn {#1} { vii } } { \bool_if_p:N \l__enumext_resume_vii_bool }
2328            { \setcounter { enumXvii } { \int_eval:n { \g__enumext_resume_vii_int } } }
2329            {
2330              \setcounter { enumX#1 }
2331                { \int_eval:n { \int_use:c { l__enumext_start_#1_int } - 1 } }
2332            }
2333          \__enumext_use_key_ref_h:
2334          \str_if_eq:nnTF {#1} { vii }
2335            {
2336              \__enumext_fake_item_vii:
2337              \bool_if:cT { l__enumext_show_length_vii_bool }
2338                { \msg_term:nnnn { enumext } { list-lengths-not-nested } { vii } { enumext* } }
2339            }
2340            {
2341              \__enumext_fake_item_viii:
2342              \bool_if:cT { l__enumext_show_length_#1_bool }
2343                { \msg_term:nnnn { enumext } { list-lengths-not-nested } { #1 } { keyans* } }
2344            }
2345        }
2346    }
2347  \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.

```
2348  \NewDocumentEnvironment{enumext}{ O{} }
2349    {
2350      \__enumext_current_env:
2351      \__enumext_safe_exec:
2352      \__enumext_parse_keys:n {#1}
2353      \__enumext_before_list:
2354      \__enumext_start_store_level:
2355      \__enumext_start_list:nn
2356        { \tl_use:c { l__enumext_label_ \__enumext_level: _tl } }
2357        {
2358          \use:c { __enumext_list_arg_two_ \__enumext_level: : }
2359          \__enumext_before_keys_exec:
2360        }
2361      \__enumext_after_args_exec:
2362    }
2363    {
2364      \__enumext_stop_list:
2365      \__enumext_stop_store_level:
2366      \__enumext_after_list:
2367    }
```

(*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.

```
2368  \cs_new_protected:Nn \__enumext_safe_exec:
```

```
2369    {
2370      \int_incr:N \l__enumext_level_int
2371      \int_compare:nNnT { \l__enumext_level_int } > { 4 }
2372        { \msg_fatal:nn { enumext } { list-too-deep } }
2373      \bool_set_true:N \l__enumext_standar_bool
2374    }
```

(*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.

```
2375  \cs_new_protected:Npn \__enumext_parse_keys:n #1
2376    {
2377      \exp_args:Ne \keys_set:nn
2378        { enumext / level-\int_use:N \l__enumext_level_int } {#1}
2379      \bool_if:NT \l__enumext_store_active_bool
2380        {
2381          \__enumext_parse_store_keys:n {#1}
2382        }
2383    }
```

(*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:.

```
2384  \cs_new_protected:Npn \__enumext_parse_store_keys:n #1
2385    {
2386      \bool_if:cF { l__enumext_store_columns_ \__enumext_level: _bool }
2387        {
2388          \regex_match:nnT { \b columns\b } {#1}
2389            {
2390              \int_set_eq:cc
2391                { l__enumext_store_columns_ \__enumext_level: _int }
2392                { l__enumext_columns_  \__enumext_level: _int }
2393              \tl_put_right:ce { l__enumext_store_opt_ \__enumext_level: _tl }
2394                {
2395                  columns = \exp_not:v { l__enumext_store_columns_ \__enumext_level: _int },
2396                }
2397            }
2398        }
2399      \bool_if:cF { l__enumext_store_columns_sep_ \__enumext_level: _bool }
2400        {
2401          \regex_match:nnT { \b columns-sep \b} {#1}
2402            {
2403              \dim_set_eq:cc
2404                { l__enumext_store_columns_sep_ \__enumext_level: _dim }
2405                { l__enumext_columns_sep_  \__enumext_level: _dim }
2406              \tl_put_right:ce { l__enumext_store_opt_ \__enumext_level: _tl }
2407                {
2408                  columns-sep = \exp_not:v { l__enumext_store_columns_sep_ \__enumext_level: _dim }
2409                }
2410            }
2411        }
2412    }
```

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

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

```
2413  \cs_new_protected:Nn \__enumext_start_store_level:
2414    {
2415      \bool_lazy_all:nT
2416        {
2417          { \bool_if_p:N \l__enumext_store_active_bool }
2418          { \bool_not_p:n { \l__enumext_keyans_env_bool } }
```

```
2419          { \bool_not_p:n { \g__enumext_starred_bool } }
2420        }
2421        {
2422          \int_compare:nNnT { \l__enumext_level_int } > { 1 }
2423            {
2424              \bool_set_true:c { l__enumext_store_upper_level_ \__enumext_level: _bool }
2425              \__enumext_store_level_open:
2426            }
2427        }
2428      \bool_lazy_all:nT
2429        {
2430          { \bool_if_p:N \l__enumext_store_active_bool }
2431          { \bool_not_p:n { \l__enumext_keyans_env_bool } }
2432          { \bool_if_p:N \g__enumext_starred_bool }
2433        }
2434        {
2435          \int_compare:nNnT { \l__enumext_level_int } > { 0 }
2436            {
2437              \bool_set_true:c { l__enumext_store_upper_level_ \__enumext_level: _bool }
2438              \__enumext_store_level_open:
2439            }
2440        }
2441    }
2442 \cs_new_protected:Nn \__enumext_stop_store_level:
2443    {
2444      \bool_if:cT { l__enumext_store_upper_level_ \__enumext_level: _bool }
2445        {
2446          \__enumext_store_level_close:
2447        }
2448    }
```

(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.

```
2449 \cs_new_protected:Nn \__enumext_before_list:
2450    {
2451      \__enumext_vspace_above:
2452      \__enumext_before_args_exec:
```

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

```
2453      \__enumext_check_ans_exec:
```

When the mini-env key is active it will set the value of the \l__enumext_minipage_right_X_dim to be the *width* of the __enumext_mini_env* environment on the *"right side"*, using this value together with the value of the \l__enumext_minipage_hsep_X_dim set by the mini-sep key, the value of \l__enumext_minipage_left_X_dim will be set, which will be the *width* of __enumext_mini_env* environment on the *"left side"*, always having a current \linewidth as *maximum width* between them.

```
2454      \dim_compare:nNnT
2455        { \dim_use:c { l__enumext_minipage_right_ \__enumext_level: _dim } } > { \c_zero_dim }
2456        {
2457          \dim_set:cn { l__enumext_minipage_left_ \__enumext_level: _dim }
2458            {
2459              \linewidth
2460              - \dim_use:c { l__enumext_minipage_right_ \__enumext_level: _dim }
2461              - \dim_use:c { l__enumext_minipage_hsep_ \__enumext_level: _dim }
2462            }
```

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*.

```
2463          \bool_set_true:c { l__enumext_minipage_active_ \__enumext_level: _bool }
2464          \int_gincr:N \g__enumext_minipage_stat_int
2465          \__enumext_mini_addvspace:
2466          \nointerlineskip\noindent
2467          \begin{__enumext_mini_env*}
```

```
2468          { \dim_use:c { l__enumext_minipage_left_ \__enumext_level: _dim } }
2469        }
2470      \__enumext_multicols_start:
2471   }
```

(*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.

```
2472 \cs_new_protected:Nn \__enumext_multicols_start:
2473   {
2474     \int_compare:nNnT
2475        { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } > { 1 }
2476        {
2477          \dim_compare:nNnT
2478            { \dim_use:c { l__enumext_columns_sep_ \__enumext_level: _dim } } = { \c_zero_dim }
2479            {
2480              \dim_set:cn { l__enumext_columns_sep_ \__enumext_level: _dim }
2481                {
2482                  ( \dim_use:c { l__enumext_labelwidth_ \__enumext_level: _dim }
2483                    + \dim_use:c { l__enumext_labelsep_ \__enumext_level: _dim }
2484                  ) / \int_use:c { l__enumext_columns_ \__enumext_level: _int }
2485                  - \dim_use:c { l__enumext_listoffset_ \__enumext_level: _dim }
2486                }
2487            }
2488          \dim_set_eq:Nc \columnsep { l__enumext_columns_sep_ \__enumext_level: _dim  }
2489          \skip_zero:N \multicolsep
2490          \int_compare:nNnT { \l__enumext_level_int } > { 1 }
2491            {
2492              \dim_zero:N \columnseprule
2493            }
```

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.

```
2494          \bool_if:cF { l__enumext_minipage_active_ \__enumext_level: _bool }
2495            {
2496              \__enumext_multi_addvspace:
2497            }
2498          \raggedcolumns
2499          \begin{multicols}{ \int_use:c { l__enumext_columns_ \__enumext_level: _int } }
2500        }
2501   }
```

(*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.

```
2502 \cs_new_protected:Nn \__enumext_multicols_stop:
2503   {
2504     \int_compare:nNnT
2505        { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } > { 1 }
2506        {
2507          \end{multicols}
2508          \bool_if:cF { l__enumext_minipage_active_ \__enumext_level: _bool }
2509            {
2510              \par\addvspace{ \skip_use:c { l__enumext_multicols_below_ \__enumext_level: _skip } }
2511            }
2512        }
```

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.

```
2513      \bool_lazy_and:nnT
2514        { \bool_if_p:N \l__enumext_check_ans_bool }
2515        { \bool_not_p:n { \g__enumext_starred_bool } }
2516        {
```

```
2517                    \bool_gset_true:N \g__enumext_check_ans_show_bool
2518                    \tl_gset:NV \g__enumext_store_name_tl \l__enumext_store_name_tl
2519                }
2520            }
```

(*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.

```
2521 \cs_new_protected:Nn \__enumext_after_list:
2522    {
2523        \bool_if:cTF { l__enumext_minipage_active_ \__enumext_level: _bool }
2524            {
2525                \int_compare:nNnT { \g__enumext_minipage_stat_int } = { 1 }
2526                    {
2527                        \msg_warning:nn { enumext } { missing-miniright }
2528                        \miniright
2529                    }
2530                \int_gzero:N \g__enumext_minipage_stat_int
2531                \end{__enumext_mini_env*}
2532                \par\addvspace { \l__enumext_minipage_after_skip }
2533            }
2534            { \__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.

```
2535        \__enumext_after_stop_list:
2536        \__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.

```
2537        \bool_set_false:N \l__enumext_standar_bool
2538        \int_gset_eq:NN \g__enumext_resume_int \value{enumXi}
2539        \int_if_exist:cT { g__enumext_resume_ \l__enumext_store_name_tl _int }
2540            {
2541                \int_gset_eq:cN
2542                    { g__enumext_resume_ \l__enumext_store_name_tl _int }
2543                    { \value{enumXi} }
2544            }
2545    }
```

(*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.

```
2546 \__enumext_after_env:nn {enumext}
2547    {
2548        \int_compare:nNnT { \l__enumext_level_int } = { 0 }
2549            {
2550                \bool_if:NT \g__enumext_check_ans_show_bool
2551                    {
2552                        \__enumext_check_ans_show:
2553                    }
2554                \bool_gset_false:N \g__enumext_standar_bool
2555                \bool_gset_false:N \g__enumext_check_ans_show_bool
2556                \tl_gclear:N \g__enumext_store_name_tl
2557            }
2558    }
```

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.

```
2559 \NewDocumentEnvironment{keyans}{ O{} }
2560    {
```

```
2561    \__enumext_keyans_safe_exec:
2562    \__enumext_keyans_parse_keys:n {#1}
2563    \__enumext_before_list_v:
2564    \__enumext_start_list:nn
2565      { \tl_use:N \l__enumext_label_v_tl }
2566      {
2567        \__enumext_list_arg_two_v:
2568        \__enumext_before_keys_exec_v:
2569      }
2570    \__enumext_after_args_exec_v:
2571  }
2572  {
2573    \__enumext_keyans_check_ans:nn { item }{ keyans }
2574    \__enumext_stop_list:
2575    \__enumext_after_list_v:
2576  }
```

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

__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.

```
2577  \cs_new_protected:Nn \__enumext_keyans_safe_exec:
2578    {
2579      \bool_if:NF \l__enumext_store_active_bool
2580        {
2581          \msg_error:nnnn { enumext } { wrong-place }{ keyans }{ save-ans }
2582        }
2583      \int_incr:N \l__enumext_keyans_level_int
2584      \bool_set_true:N \l__enumext_keyans_env_bool
2585      % Set false for interfering with enumext nested in keyans (yes, its possible and crayze)
2586      \bool_set_false:N \l__enumext_store_active_bool
2587      \int_compare:nNnT { \l__enumext_keyans_level_int } > { 1 }
2588        {
2589          \msg_error:nn { enumext } { keyans-nested }
2590        }
2591      \int_compare:nNnT { \l__enumext_level_int } > { 1 }
2592        {
2593          \msg_error:nn { enumext } { keyans-wrong-level }
2594        }
2595    }
```

(*End of definition for* __enumext_keyans_safe_exec:.)

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

```
2596  \cs_new_protected:Npn \__enumext_keyans_parse_keys:n #1
2597    {
2598      \keys_set:nn { enumext / keyans } {#1}
2599    }
```

(*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.

```
2600  \cs_new_protected:Nn \__enumext_before_list_v:
2601    {
2602      \__enumext_vspace_above_v:
2603      \__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.

```
2604      \dim_compare:nNnT { \l__enumext_minipage_right_v_dim } > { \c_zero_dim }
2605        {
2606          \dim_set:Nn \l__enumext_minipage_left_v_dim
2607            {
2608              \linewidth - \l__enumext_minipage_right_v_dim - \l__enumext_minipage_hsep_v_dim
2609            }
```

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*.

```
2610        \bool_set_true:N \l__enumext_minipage_active_v_bool
2611        \int_gincr:N \g__enumext_minipage_stat_int
2612        \__enumext_keyans_mini_addvspace:
2613        \nointerlineskip\noindent
2614        \begin{__enumext_mini_env*}{ \l__enumext_minipage_left_v_dim }
2615      }
```

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

```
2616      \__enumext_keyans_multicols_start:
2617    }
```

(*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.

```
2618  \cs_new_protected:Nn \__enumext_keyans_multicols_start:
2619    {
2620      \int_compare:nNnT { \l__enumext_columns_v_int } > { 1 }
2621        {
```

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

```
2622          \dim_compare:nNnT { \l__enumext_columns_sep_v_dim } = { \c_zero_dim }
2623            {
2624              \dim_set:Nn \l__enumext_columns_sep_v_dim
2625                {
2626                  (
2627                    \l__enumext_labelwidth_v_dim + \l__enumext_labelsep_v_dim
2628                  ) / \l__enumext_columns_v_int
2629                  - \l__enumext_listoffset_v_dim
2630                }
2631            }
2632          \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).

```
2633          \skip_zero:N \multicolsep
2634          \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.

```
2635          \bool_if:NF \l__enumext_minipage_active_v_bool
2636            {
2637              \__enumext_keyans_multi_addvspace:
2638            }
2639          \raggedcolumns
2640          \begin{multicols}{ \l__enumext_columns_v_int }
2641        }
2642    }
```

(*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.

```
2643  \cs_new_protected:Nn \__enumext_keyans_multicols_stop:
2644    {
2645      \int_compare:nNnT { \l__enumext_columns_v_int } > { 1 }
2646        {
2647          \end{multicols}
2648          \bool_if:NF \l__enumext_minipage_active_v_bool
2649            {
2650              \par\addvspace{ \l__enumext_multicols_below_v_skip }
2651            }
2652        }
2653    }
```

(*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.

```
2654 \cs_new_protected:Nn \__enumext_after_list_v:
2655   {
2656     \bool_if:NTF \l__enumext_minipage_active_v_bool
2657       {
2658         \int_compare:nNnT { \g__enumext_minipage_stat_int } = { 1 }
2659           {
2660             \msg_warning:nn { enumext } { missing-miniright }
2661             \miniright
2662           }
2663         \int_gzero:N \g__enumext_minipage_stat_int
2664         \end{__enumext_mini_env*}
2665         \par\addvspace{ \l__enumext_minipage_after_skip }
2666       }
2667       { \__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.

```
2668     \bool_set_false:N \l__enumext_keyans_env_bool
2669     \__enumext_after_stop_list_v:
2670     \__enumext_vspace_below_v:
2671   }
```

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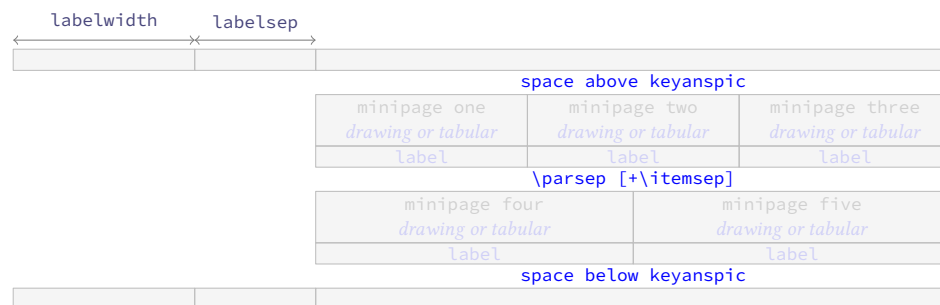
Figure 12: Representation of the keyanspic spacing in enumext.

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.

```
2672 \NewDocumentCommand \anspic { s o +m }
2673   {
```

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

```
2674     \bool_if:NF \l__enumext_store_active_bool
2675       {
2676         \msg_error:nnnn { enumext } { wrong-place }{ keyanspic }{ save-ans }
2677       }
2678     \int_compare:nNnT { \l__enumext_level_int } > { 1 }
2679       {
```

```
2680        \msg_error:nn { enumext } { keyanspic-wrong-level }
2681        }
2682      \int_compare:nNnT { \l__enumext_keyans_level_int } = { 1 }
2683        {
2684          \msg_error:nnnn { enumext } { command-wrong-place }{ anspic }{ keyans }
2685        }
```

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.

```
2686      \seq_put_right:Nn \l__enumext_keyans_pic_body_seq
2687        {
2688          \__enumext_keyans_anspic_code:nnn { #1 } { #2 } { #3 }
2689        }
2690    }
```

(*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.

```
2691  \cs_new_protected:Nn \__enumext_keyans_anspic_code:nnn
2692    {
2693      \stepcounter { enumXvi }
2694      #3 \\
2695      \bool_if:nT { #1 }
2696        {
2697          \__enumext_keyans_addto_prop:n { #2 }
2698          \__enumext_keyans_store_ref:
2699          \__enumext_keyans_addto_seq:n { #2 }
2700          \bool_lazy_or:nnT
2701            { \bool_if_p:N \l__enumext_show_answer_bool }
2702            { \bool_if_p:N \l__enumext_show_position_bool }
2703            {
2704              \tl_set_eq:NN \l__enumext_label_v_tl \l__enumext_label_vi_tl
2705              \__enumext_keyans_show_left:n { #2 }
2706              \tl_set_eq:NN \l__enumext_label_vi_tl \l__enumext_label_v_tl
2707            }
2708        }
2709      \tl_use:N \l__enumext_label_font_style_v_tl
2710      \__enumext_wrapper_label_v:n { \l__enumext_label_vi_tl } \__enumext_keyans_show_item_opt:
2711    }
```

(*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.

```
2712  \NewDocumentEnvironment{keyanspic}{ o }
2713    {
2714      \__enumext_keyans_pic_safe_exec:
2715      \__enumext_start_list:nn
2716        { }
2717        {
2718          \__enumext_keyans_pic_arg_two:
2719        }
```

We apply the "adjusted" vertical spacing above the environment

```
2720        \vspace { \l__enumext_keyans_pic_above_skip }
2721    }
```

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.

```
2722    {
2723      \tl_if_novalue:nTF { #1 }
2724        {
2725          \__enumext_keyans_pic_do:e { \seq_count:N \l__enumext_keyans_pic_body_seq }
2726        }
2727        { \__enumext_keyans_pic_do:n { #1 } }
2728      \__enumext_stop_list:
2729      \__enumext_keyans_check_ans:nn { anspic } { keyanspic }
```

```
2730      \setcounter { enumXvi } { 0 }
2731      \vspace { \l__enumext_topsep_v_skip }
2732      %\bool_set_false:N \l__enumext_store_active_bool
2733    }
```

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

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

```
2734 \cs_new_protected:Nn \__enumext_keyans_pic_safe_exec:
2735    {
2736      \int_incr:N \l__enumext_keyans_pic_level_int
2737      \int_compare:nNnT { \l__enumext_keyans_pic_level_int } > { 1 }
2738        {
2739          \msg_error:nn { enumext } { keyanspic-nested }
2740        }
2741    }
```

(*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.

```
2742 \cs_new_protected:Npn \__enumext_keyans_pic_skip_abs:N #1
2743    {
2744      \dim_compare:nNnT { #1 } < { 0pt }
2745        { \skip_set:Nn #1 { -#1 } }
2746    }
```

(*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.

```
2747 \cs_new_protected:Nn \__enumext_keyans_pic_arg_two:
2748    {
```

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.

```
2749      \bool_set_false:N \l__enumext_leftmargin_tmp_v_bool
2750      \__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.

```
2751      \skip_add:Nn \parsep { \itemsep }
2752      \dim_add:Nn  \leftmargin { -\labelwidth - \labelsep }
2753      \dim_zero:N  \labelwidth
2754      \dim_zero:N  \listparindent
2755      \dim_zero:N  \labelsep
2756      \skip_zero:N \partopsep
2757      \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.

```
2758      \__enumext_keyans_pic_skip_abs:N \parsep
2759      \skip_set:Nn \l__enumext_keyans_pic_above_skip
2760        {
2761          \box_dp:N \strutbox
2762          + \l__enumext_topsep_v_skip
2763          - \parsep
2764        }
2765      \__enumext_item_std:w \scan_stop:
2766    }
```

(*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.

```
2767 \cs_new_protected:Nn \__enumext_keyans_pic_do:n
2768   {
2769     \clist_map_function:nN { #1 } \__enumext_keyans_pic_row:n
2770   }
2771 \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.

```
2772 \cs_new_protected:Nn \__enumext_keyans_pic_row:n
2773   {
2774     \dim_set:Nn \l__enumext_keyans_pic_width_dim { \linewidth / #1 }
2775     \int_set:Nn \l__enumext_keyans_pic_above_int { \l__enumext_keyans_pic_below_int }
2776     \int_set:Nn \l__enumext_keyans_pic_below_int { \l__enumext_keyans_pic_above_int + #1 }
2777     \int_step_inline:nnn
2778       { \l__enumext_keyans_pic_above_int + 1 }
2779       { \l__enumext_keyans_pic_below_int }
2780       {
2781         \__enumext_minipage:w [ b ]{ \l__enumext_keyans_pic_width_dim }
2782           \centering
2783           \seq_item:Nn \l__enumext_keyans_pic_body_seq { ##1 }
2784         \__enumext_endminipage:
2785       }
2786     \par
2787   }
```

(*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.

```
2788 \cs_new_protected:Nn \__enumext_starred_columns_set_vii:
2789   {
2790     \dim_compare:nNnT { \l__enumext_columns_sep_vii_dim } = { \c_zero_dim }
2791       {
2792         \dim_set:Nn \l__enumext_columns_sep_vii_dim
2793           {
2794             ( \l__enumext_labelwidth_vii_dim + \l__enumext_labelsep_vii_dim )
2795             / \l__enumext_columns_vii_int
2796           }
2797       }
2798     \int_set:Nn \l__enumext_tmpa_vii_int { \l__enumext_columns_vii_int - \c_one_int }
2799     \dim_set:Nn \l__enumext_item_width_vii_dim
2800       {
2801         ( \linewidth - \l__enumext_columns_sep_vii_dim * \l__enumext_tmpa_vii_int )
2802         / \l__enumext_columns_vii_int - \l__enumext_labelwidth_vii_dim
2803         - \l__enumext_labelsep_vii_dim
2804       }
2805     \dim_zero_new:N \itemwidth
2806   }
```

(*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`.

```
2807  \cs_new_protected:Npn \__enumext_starred_joined_item_vii:n #1
2808    {
2809      \int_set:Nn \l__enumext_joined_item_vii_int {#1}
2810      \int_compare:nNnT { \l__enumext_joined_item_vii_int } > { \l__enumext_columns_vii_int }
2811        {
2812          \msg_warning:nnee { enumext } { item-joined }
2813            { \int_use:N \l__enumext_joined_item_vii_int }
2814            { \int_use:N \l__enumext_columns_vii_int }
2815          \int_set:Nn \l__enumext_joined_item_vii_int
2816            {
2817              \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + \c_one_int
2818            }
2819        }
2820      \int_compare:nNnT
2821        { \l__enumext_joined_item_vii_int }
2822        >
2823        { \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + \c_one_int }
2824        {
2825          \msg_warning:nnee { enumext } { item-joined-columns }
2826            { \int_use:N \l__enumext_joined_item_vii_int }
2827            {
2828              \int_eval:n
2829                { \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + \c_one_int }
2830            }
2831          \int_set:Nn \l__enumext_joined_item_vii_int
2832            {
2833              \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + \c_one_int
2834            }
2835        }
```

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

```
2836      \int_compare:nNnTF { \l__enumext_joined_item_vii_int } > { \c_one_int }
2837        {
2838          \int_set_eq:NN \l__enumext_joined_item_aux_vii_int \l__enumext_joined_item_vii_int
2839          \int_decr:N \l__enumext_joined_item_aux_vii_int
2840          \int_add:Nn \l__enumext_item_column_pos_vii_int { \l__enumext_joined_item_aux_vii_int }
2841          \int_gadd:Nn \g__enumext_item_count_all_vii_int { \l__enumext_joined_item_aux_vii_int }
2842          \dim_set:Nn \l__enumext_joined_width_vii_dim
2843            {
2844              \l__enumext_item_width_vii_dim * \l__enumext_joined_item_vii_int
2845              + (   \l__enumext_labelwidth_vii_dim + \l__enumext_labelsep_vii_dim
2846                  + \l__enumext_columns_sep_vii_dim
2847                )*\l__enumext_joined_item_aux_vii_int
2848            }
2849          \dim_set_eq:NN \itemwidth \l__enumext_joined_width_vii_dim
2850        }
2851        {
2852          \dim_set_eq:NN \l__enumext_joined_width_vii_dim \l__enumext_item_width_vii_dim
2853          \dim_set_eq:NN \itemwidth \l__enumext_item_width_vii_dim
2854        }
2855    }
```

(*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`.

```
2856  \cs_new_protected:Nn \__enumext_start_mini_vii:
2857    {
2858      \dim_compare:nNnT { \l__enumext_minipage_right_vii_dim } > { \c_zero_dim }
2859        {
2860          \dim_set:Nn \l__enumext_minipage_left_vii_dim
2861            {
2862              \linewidth
2863              - \l__enumext_minipage_right_vii_dim
2864              - \l__enumext_minipage_hsep_vii_dim
2865            }
2866          \bool_set_true:N \l__enumext_minipage_active_vii_bool
```

```
2867        \dim_gset_eq:NN
2868          \g__enumext_minipage_right_vii_dim
2869          \l__enumext_minipage_right_vii_dim
2870        \__enumext_mini_addvspace_vii:
2871        \nointerlineskip\noindent
2872        \begin{__enumext_mini_env*}{ \l__enumext_minipage_left_vii_dim }
2873      }
2874    }
```

(*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"*.

```
2875  \cs_new_protected:Nn \__enumext_stop_mini_vii:
2876    {
2877      \bool_if:NT \l__enumext_minipage_active_vii_bool
2878        {
2879          \end{__enumext_mini_env*}
2880          \hfill
2881          \bool_gset_true:N \g__enumext_minipage_active_vii_bool
2882        }
2883    }
```

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"*.

```
2884  \__enumext_after_env:nn {enumext*}
2885    {
2886      \bool_if:NT \g__enumext_minipage_active_vii_bool
2887        {
2888          \begin{__enumext_mini_env*}{ \g__enumext_minipage_right_vii_dim }
2889            \par\addvspace { \g__enumext_minipage_right_skip }
2890            \bool_if:NF \g__enumext_minipage_center_vii_bool
2891              {
2892                \centering
2893              }
2894            \tl_use:N \g__enumext_miniright_code_vii_tl % the code
2895          \end{__enumext_mini_env*}
2896          \par\addvspace{ \g__enumext_minipage_after_skip }
2897        }
2898      \bool_gset_false:N \g__enumext_minipage_active_vii_bool
2899      \bool_gset_true:N \g__enumext_minipage_center_vii_bool
2900      \tl_gclear:N \g__enumext_miniright_code_vii_tl
2901      \dim_gzero:N \g__enumext_minipage_right_vii_dim
2902    }
```

(*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.

```
2903  \NewDocumentEnvironment{enumext*}{ o }
2904    {
2905      \__enumext_current_env:
2906      \__enumext_safe_exec_vii:
2907      \__enumext_parse_keys_vii:n {#1}
2908      \__enumext_before_list_vii:
2909      \__enumext_start_store_level_vii:
2910      \__enumext_start_list:nn { }
2911        {
2912          \__enumext_list_arg_two_vii:
2913          \__enumext_before_keys_exec_vii:
2914        }
2915      \__enumext_starred_columns_set_vii:
2916      \item[] \scan_stop:
2917      \cs_set_eq:NN \__enumext_stop_item_tmp_vii: \noindent
2918      \cs_set_eq:NN \item \__enumext_start_item_tmp_vii:
2919    }
2920    {
2921      \__enumext_stop_item_tmp_vii:
```

```
2922        \__enumext_remove_extra_parsep_vii:
2923        \__enumext_stop_list:
2924        \__enumext_stop_store_level_vii:
2925        \__enumext_after_list_vii:
2926      }
```

(*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.

```
2927  \cs_new_protected:Nn \__enumext_safe_exec_vii:
2928    {
2929      \int_incr:N \l__enumext_level_h_int
2930      \int_compare:nNnT { \l__enumext_level_h_int } > { 1 }
2931        {
2932          \msg_error:nn { enumext } { nested }
2933        }
2934      \bool_set_true:N \l__enumext_starred_bool
2935    }
```

(*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.

```
2936  \cs_new_protected:Npn \__enumext_parse_keys_vii:n #1
2937    {
2938      \tl_if_novalue:nF {#1}
2939        {
2940          \keys_set:nn { enumext / enumext* } {#1}
2941          \bool_if:NT \l__enumext_store_active_bool
2942            {
2943              \__enumext_parse_store_keys_vii:n {#1}
2944            }
2945        }
2946    }
```

(*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:.

```
2947  \cs_new_protected:Npn \__enumext_parse_store_keys_vii:n #1
2948    {
2949      \bool_if:NF \l__enumext_store_columns_vii_bool
2950        {
2951          \regex_match:nnT { \b columns\b } {#1}
2952            {
2953              \int_set_eq:NN
2954                \l__enumext_store_columns_vii_int
2955                \l__enumext_columns_vii_int
2956              \tl_put_right:Ne \l__enumext_store_opt_vii_tl
2957                {
2958                  columns = \exp_not:V \l__enumext_store_columns_vii_int ,
2959                }
2960            }
2961        }
2962      \bool_if:NF \l__enumext_store_columns_sep_vii_bool
2963        {
2964          \regex_match:nnT { \b columns-sep \b} {#1}
2965            {
2966              \dim_set_eq:NN
2967                \l__enumext_store_columns_sep_vii_dim
2968                \l__enumext_columns_sep_vii_dim
2969              \tl_put_right:Ne \l__enumext_store_opt_vii_tl
2970                {
2971                  columns-sep = \exp_not:V \l__enumext_store_columns_sep_vii_dim,
2972                }
```

```
2973          }
2974        }
2975    }
```

(*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.

```
2976 \cs_new_protected:Nn \__enumext_before_list_vii:
2977    {
2978      \__enumext_vspace_above_vii:
2979      \__enumext_check_ans_exec: % need by chek-ans
2980      \__enumext_before_args_exec_vii:
2981      \__enumext_start_mini_vii:
2982    }
```

(*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.

```
2983 \cs_new_protected:Nn \__enumext_after_list_vii:
2984    {
2985      \__enumext_stop_mini_vii:
2986      \__enumext_after_stop_list_vii:
2987      \__enumext_vspace_below_vii:
2988      \int_gset_eq:NN \g__enumext_resume_vii_int \value{enumXvii}
2989      \int_if_exist:cT { g__enumext_resume_ \l__enumext_store_name_tl _int }
2990        {
2991          \int_gset_eq:cN
2992            { g__enumext_resume_ \l__enumext_store_name_tl _int }
2993            { \value{enumXvii} }
2994        }
2995      \bool_lazy_and:nnT
2996        { \bool_if_p:N \g__enumext_starred_bool }
2997        { \bool_if_p:N \l__enumext_check_ans_bool }
2998        {
2999          \bool_gset_true:N \g__enumext_check_ans_show_h_bool
3000          \tl_gset:NV \g__enumext_store_name_tl \l__enumext_store_name_tl
3001        }
3002      %\bool_gset_false:N \g__enumext_starred_bool
3003      \bool_set_false:N \l__enumext_starred_bool
3004    }
```

(*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.

```
3005 \cs_new_protected:Nn \__enumext_start_store_level_vii:
3006    {
3007      \bool_if:NT \l__enumext_store_active_bool
3008        {
3009          \int_compare:nNnT { \l__enumext_level_int } > { \c_zero_int }
3010            {
3011              \__enumext_store_level_open_vii:
3012            }
3013        }
3014    }
3015 \cs_new_protected:Nn \__enumext_stop_store_level_vii:
3016    {
3017      \bool_if:NT \l__enumext_store_active_bool
3018        {
3019          \int_compare:nNnT { \l__enumext_level_int } > { \c_zero_int }
3020            {
3021              \__enumext_store_level_close_vii:
```

```
3022                    }
3023              }
3024       }
```

(*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.

```
3025 \cs_new_protected_nopar:Nn \__enumext_start_item_tmp_vii:
3026   {
3027     \__enumext_stop_item_tmp_vii:
3028     \int_incr:N \l__enumext_item_column_pos_vii_int
3029     \int_gincr:N \g__enumext_item_count_all_vii_int
3030     \__enumext_item_peek_args_vii:
3031   }
```

(*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).

```
3032 \cs_new_protected:Nn \__enumext_item_peek_args_vii:
3033   {
3034     \peek_meaning:NTF (
3035       { \__enumext_joined_item_vii:w }
3036       { \__enumext_joined_item_vii:w (1) }
3037   }
```

(*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.

```
3038 \cs_new_protected:Npn \__enumext_joined_item_vii:w (#1)
3039   {
3040     \__enumext_starred_joined_item_vii:n {#1}
3041     \peek_meaning_remove:NTF *
3042       { \__enumext_starred_item_vii:w  }
3043       { \__enumext_standard_item_vii:w }
3044   }
```

(*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].

```
3045 \cs_new_protected:Npn \__enumext_standard_item_vii:w
3046   {
3047     \bool_set_false:N \l__enumext_item_starred_vii_bool
3048       \peek_meaning:NTF [
3049         {
3050           \bool_set_eq:NN
3051             \l__enumext_wrap_label_vii_bool
3052             \l__enumext_wrap_label_opt_vii_bool
3053           \__enumext_start_item_vii:w
3054         }
3055         {
3056           \bool_set_true:N \l__enumext_wrap_label_vii_bool
3057           \legacy_if_set_true:n { @noitemarg }
```

```
3058                    \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
3059              }
3060        }
```

(*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*⟩].

```
3061  \cs_new_protected:Npn \__enumext_starred_item_vii:w
3062    {
3063      \bool_set_true:N \l__enumext_item_starred_vii_bool
3064      \bool_set_true:N \l__enumext_wrap_label_vii_bool
3065      \peek_meaning:NTF [
3066        { \__enumext_starred_item_vii_aux_i:w }
3067        { \__enumext_starred_item_vii_aux_ii:w }
3068    }
3069  \cs_new_protected:Npn \__enumext_starred_item_vii_aux_i:w [#1]
3070    {
3071      \tl_gset:Nn \g__enumext_item_symbol_aux_vii_tl {#1}
3072      \__enumext_starred_item_vii_aux_ii:w
3073    }
3074  \cs_new_protected:Npn \__enumext_starred_item_vii_aux_ii:w
3075    {
3076      \peek_meaning:NTF [
3077        { \__enumext_starred_item_vii_aux_iii:w }
3078        {
3079          \dim_set_eq:NN
3080            \l__enumext_item_symbol_sep_vii_dim
3081            \l__enumext_labelsep_vii_dim
3082          \legacy_if_set_true:n { @noitemarg }
3083          \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
3084        }
3085    }
3086  \cs_new_protected:Npn \__enumext_starred_item_vii_aux_iii:w [#1]
3087    {
3088      \dim_set:Nn \l__enumext_item_symbol_sep_vii_dim {#1}
3089      \legacy_if_set_true:n { @noitemarg }
3090      \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
3091    }
```

(*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.

```
3092  \cs_new_protected_nopar:Npn \__enumext_start_item_vii:w [#1]
3093    {
3094      \cs_set_eq:NN \__enumext_stop_item_tmp_vii: \__enumext_stop_item_vii:
3095      \legacy_if:nT { @noitemarg }
3096        {
3097          \legacy_if_set_false:n { @noitemarg }
3098          \legacy_if:nT { @nmbrlist }
3099            {
3100              \bool_if:NT \l__enumext_hyperref_bool
3101                {
3102                  \legacy_if_set_true:n { @hyper@item }
3103                }
3104              \refstepcounter{enumXvii}
3105              \bool_if:NT \l__enumext_check_ans_bool
3106                {
3107                  \int_gincr:N \g__enumext_count_item_number_int
3108                }
3109            }
3110        }
```

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.

```
3111      \group_begin:
3112        \lrbox{ \l__enumext_item_text_vii_box }
3113          \bool_if:NF \l__enumext_footnotes_key_bool
3114            {
3115              \__enumext_renew_footnote:
3116            }
3117          \bool_if:NT \l__enumext_item_starred_vii_bool
3118            {
3119              \tl_if_blank:VT \g__enumext_item_symbol_aux_vii_tl
3120                {
3121                  \tl_gset_eq:NN
3122                    \g__enumext_item_symbol_aux_vii_tl \l__enumext_item_symbol_vii_tl
3123                }
3124              \mode_leave_vertical:
3125              \skip_horizontal:n { -\l__enumext_item_symbol_sep_vii_dim }
3126              \makebox[ 0pt ][ r ]{ \g__enumext_item_symbol_aux_vii_tl }
3127              \skip_horizontal:N \l__enumext_item_symbol_sep_vii_dim
3128              \tl_gclear:N \g__enumext_item_symbol_aux_vii_tl
3129            }
3130          \group_begin:
3131            \tl_use:N \l__enumext_label_font_style_vii_tl
3132            \bool_if:NTF \l__enumext_wrap_label_vii_bool
3133              {
3134                \makebox[ \l__enumext_labelwidth_vii_dim ][ \l__enumext_align_label_vii_str ]
3135                  { \__enumext_wrapper_label_vii:n {#1} }
3136              }
3137              {
3138                \makebox[ \l__enumext_labelwidth_vii_dim ][ \l__enumext_align_label_vii_str ]{ #1 }
3139              }
3140          \group_end:
3141          \skip_horizontal:N \l__enumext_labelsep_vii_dim
3142          \tl_use:N \l__enumext_after_list_args_vii_tl
3143          \__enumext_minipage:w [ t ]{ \l__enumext_joined_width_vii_dim  }
3144            \skip_set_eq:NN \parindent \l__enumext_listparindent_vii_dim
3145            \skip_set_eq:NN \parskip \l__enumext_parsep_vii_skip
3146            \tl_use:N \l__enumext_fake_item_indent_vii_tl
3147      }
```

(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.

```
3148  \cs_new_protected_nopar:Nn \__enumext_stop_item_vii:
3149    {
3150          \__enumext_endminipage:
3151        \endlrbox
3152      \group_end:
3153      \box_set_wd:Nn \l__enumext_item_text_vii_box
3154        {
3155          \l__enumext_joined_width_vii_dim
3156          + \l__enumext_labelwidth_vii_dim
3157          + \l__enumext_labelsep_vii_dim
3158        }
3159      \int_set:Nn \hbadness { 10000 }
3160      \box_use:N \l__enumext_item_text_vii_box
3161      \bool_if:NF \l__enumext_footnotes_key_bool
3162        {
3163          \__enumext_print_footnote:
3164        }
3165      \int_compare:nNnTF { \l__enumext_item_column_pos_vii_int } = { \l__enumext_columns_vii_int }
3166        {
3167          \par\noindent
```

```
3168         \int_zero:N \l__enumext_item_column_pos_vii_int
3169       }
3170     { \hspace{ \l__enumext_columns_sep_vii_dim } }
3171   }
```

(*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.

```
3172 \cs_new_protected:Nn \__enumext_remove_extra_parsep_vii:
3173   {
3174     \int_compare:nNnT
3175       {
3176         \int_mod:nn {  \g__enumext_item_count_all_vii_int } { \l__enumext_columns_vii_int }
3177       }
3178       =
3179       { \c_zero_int }
3180       {
3181         \par
3182         \vspace{ -\l__enumext_itemsep_vii_skip }
3183         \int_gzero:N \g__enumext_item_count_all_vii_int
3184       }
3185   }
```

(*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.

```
3186 \__enumext_after_env:nn {enumext*}
3187   {
3188     \int_compare:nNnT { \l__enumext_level_int } = { 0 }
3189       {
3190         \bool_if:NT \g__enumext_check_ans_show_h_bool
3191           {
3192             \__enumext_check_ans_show:
3193           }
3194         \bool_gset_false:N \g__enumext_starred_bool
3195         \bool_gset_false:N \g__enumext_check_ans_show_h_bool
3196         \tl_gclear:N \g__enumext_store_name_tl
3197       }
3198   }
```

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.

```
3199 \cs_new_protected:Nn \__enumext_starred_columns_set_viii:
3200   {
3201     \dim_compare:nNnT { \l__enumext_columns_sep_viii_dim } = { \c_zero_dim }
3202       {
3203         \dim_set:Nn \l__enumext_columns_sep_viii_dim
3204           {
3205             ( \l__enumext_labelwidth_viii_dim + \l__enumext_labelsep_viii_dim )
3206             / \l__enumext_columns_viii_int
3207           }
3208       }
3209     \int_set:Nn \l__enumext_tmpa_viii_int { \l__enumext_columns_viii_int - \c_one_int }
3210     \dim_set:Nn \l__enumext_item_width_viii_dim
3211       {
3212         ( \linewidth - \l__enumext_columns_sep_viii_dim * \l__enumext_tmpa_viii_int )
3213         / \l__enumext_columns_viii_int - \l__enumext_labelwidth_viii_dim
3214         - \l__enumext_labelsep_viii_dim
3215       }
3216     \dim_zero_new:N \itemwidth
3217   }
```

(*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`.

```
3218  \cs_new_protected:Npn \__enumext_starred_joined_item_viii:n #1
3219    {
3220      \int_set:Nn \l__enumext_joined_item_viii_int {#1}
3221      \int_compare:nNnT { \l__enumext_joined_item_viii_int } > { \l__enumext_columns_viii_int }
3222        {
3223          \msg_warning:nnee { enumext } { item-joined }
3224            { \int_use:N \l__enumext_joined_item_viii_int }
3225            { \int_use:N \l__enumext_columns_viii_int }
3226          \int_set:Nn \l__enumext_joined_item_viii_int
3227            {
3228              \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + \c_one_int
3229            }
3230        }
3231      \int_compare:nNnT
3232        { \l__enumext_joined_item_viii_int }
3233        >
3234        { \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + \c_one_int }
3235        {
3236          \msg_warning:nnee { enumext } { item-joined-columns }
3237            { \int_use:N \l__enumext_joined_item_viii_int }
3238            {
3239              \int_eval:n
3240                { \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + \c_one_int }
3241            }
3242          \int_set:Nn \l__enumext_joined_item_viii_int
3243            {
3244              \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + \c_one_int
3245            }
3246        }
3247      \int_compare:nNnTF { \l__enumext_joined_item_viii_int } > { \c_one_int }
3248        {
3249          \int_set_eq:NN \l__enumext_joined_item_aux_viii_int \l__enumext_joined_item_viii_int
3250          \int_decr:N \l__enumext_joined_item_aux_viii_int
3251          \int_add:Nn \l__enumext_item_column_pos_viii_int { \l__enumext_joined_item_aux_viii_int }
3252          \int_gadd:Nn \g__enumext_item_count_all_viii_int { \l__enumext_joined_item_aux_viii_int }
3253          \dim_set:Nn \l__enumext_joined_width_viii_dim
3254            {
3255              \l__enumext_item_width_viii_dim * \l__enumext_joined_item_viii_int
3256              + (  \l__enumext_labelwidth_viii_dim + \l__enumext_labelsep_viii_dim
3257                + \l__enumext_columns_sep_viii_dim
3258              )*\l__enumext_joined_item_aux_viii_int
3259            }
3260          \dim_set_eq:NN \itemwidth \l__enumext_joined_width_viii_dim
3261        }
3262        {
3263          \dim_set_eq:NN \l__enumext_joined_width_viii_dim \l__enumext_item_width_viii_dim
3264          \dim_set_eq:NN \itemwidth \l__enumext_item_width_viii_dim
3265        }
3266    }
```

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

__enumext_start_mini_viii:
__enumext_stop_mini_viii:

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

```
3267  \cs_new_protected:Nn \__enumext_start_mini_viii:
3268    {
3269      \dim_compare:nNnT { \l__enumext_minipage_right_viii_dim } > { \c_zero_dim }
3270        {
3271          \dim_set:Nn \l__enumext_minipage_left_viii_dim
3272            {
3273              \linewidth
3274              - \l__enumext_minipage_right_viii_dim
3275              - \l__enumext_minipage_hsep_viii_dim
3276            }
3277          \bool_set_true:N \l__enumext_minipage_active_viii_bool
3278          \dim_gset_eq:NN
3279            \g__enumext_minipage_right_viii_dim
3280            \l__enumext_minipage_right_viii_dim
3281          \__enumext_mini_addvspace_viii:
```

```
3282        \nointerlineskip\noindent
3283        \begin{__enumext_mini_env*}{ \l__enumext_minipage_left_viii_dim }
3284      }
3285    }
3286 \cs_new_protected:Nn \__enumext_stop_mini_viii:
3287   {
3288     \bool_if:NT \l__enumext_minipage_active_viii_bool
3289       {
3290         \end{__enumext_mini_env*}
3291         \hfill
3292         \bool_gset_true:N \g__enumext_minipage_active_viii_bool
3293       }
3294   }
3295 \__enumext_after_env:nn {keyans*}
3296   {
3297     \bool_if:NT \g__enumext_minipage_active_viii_bool
3298       {
3299         \begin{__enumext_mini_env*}{ \g__enumext_minipage_right_viii_dim }
3300           \par\addvspace { \g__enumext_minipage_right_skip }
3301           \bool_if:NF \g__enumext_minipage_center_viii_bool
3302             {
3303               \centering
3304             }
3305           \tl_use:N \g__enumext_miniright_code_viii_tl % the code
3306         \end{__enumext_mini_env*}
3307         \par\addvspace{ \g__enumext_minipage_after_skip }
3308       }
3309     \bool_gset_false:N \g__enumext_minipage_active_viii_bool
3310     \bool_gset_true:N \g__enumext_minipage_center_viii_bool
3311     \tl_gclear:N \g__enumext_miniright_code_viii_tl
3312     \dim_gzero:N \g__enumext_minipage_right_viii_dim
3313   }
```

(*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.

```
3314 \NewDocumentEnvironment{keyans*}{ o }
3315   {
3316     \__enumext_safe_exec_viii:
3317     \__enumext_parse_keys_viii:n {#1}
3318     \__enumext_before_list_viii:
3319     \__enumext_start_list:nn { }
3320       {
3321         \__enumext_list_arg_two_viii:
3322         \__enumext_before_keys_exec_viii:
3323       }
3324     \__enumext_starred_columns_set_viii:
3325     \item[] \scan_stop:
3326     \cs_set_eq:NN \__enumext_stop_item_tmp_viii: \noindent
3327     \cs_set_eq:NN \item \__enumext_start_item_tmp_viii:
3328   }
3329   {
3330     \__enumext_stop_item_tmp_viii:
3331     \__enumext_remove_extra_parsep_viii:
3332     \__enumext_keyans_check_ans:nn { item }{ keyans* }
3333     \__enumext_stop_list:
3334     \__enumext_after_list_viii:
3335   }
```

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

__enumext_safe_exec_viii: First check the maximum nesting level for the keyans* environment.

```
3336 \cs_new_protected:Nn \__enumext_safe_exec_viii:
3337   {
3338     \int_incr:N \l__enumext_keyans_level_h_int
3339     \int_compare:nNnT { \l__enumext_keyans_level_h_int } > { 1 }
3340       {
3341         \msg_error:nn { enumext } { nested }
3342       }
```

```
3343        % Set false for interfering with enumext nested in keyans* (yes, its possible and crayze)
3344        \bool_set_false:N \l__enumext_store_active_bool
3345        \int_compare:nNnT { \l__enumext_level_int } > { 1 }
3346          {
3347            \msg_error:nn { enumext } { keyans-wrong-level }
3348          }
3349      }
```

(*End of definition for* __enumext_safe_exec_viii:.)

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

```
3350 \cs_new_protected:Npn \__enumext_parse_keys_viii:n #1
3351   {
3352     \tl_if_novalue:nF {#1}
3353       {
3354         \keys_set:nn { enumext / keyans* } {#1}
3355       }
3356   }
```

(*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.

```
3357 \cs_new_protected:Nn \__enumext_before_list_viii:
3358   {
3359     \__enumext_vspace_above_viii:
3360     \__enumext_before_args_exec_viii:
3361     \__enumext_start_mini_viii:
3362   }
```

(*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.

```
3363 \cs_new_protected:Nn \__enumext_after_list_viii:
3364   {
3365     \__enumext_stop_mini_viii:
3366     \__enumext_after_stop_list_viii:
3367     \__enumext_vspace_below_viii:
3368   }
```

(*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.

```
3369 \cs_new_protected_nopar:Nn \__enumext_start_item_tmp_viii:
3370   {
3371     \__enumext_stop_item_tmp_viii:
3372     \int_incr:N \l__enumext_item_column_pos_viii_int
3373     \int_gincr:N \g__enumext_item_count_all_viii_int
3374     \__enumext_item_peek_args_viii:
3375   }
```

(*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).

```
3376  \cs_new_protected:Nn \__enumext_item_peek_args_viii:
3377    {
3378      \peek_meaning:NTF (
3379        { \__enumext_joined_item_viii:w }
3380        { \__enumext_joined_item_viii:w (1) }
3381    }
```

(*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.

```
3382  \cs_new_protected:Npn \__enumext_joined_item_viii:w (#1)
3383    {
3384      \__enumext_starred_joined_item_viii:n {#1}
3385      \peek_meaning_remove:NTF *
3386        { \__enumext_starred_item_viii:w  }
3387        { \__enumext_standard_item_viii:w }
3388    }
```

(*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].

```
3389  \cs_new_protected:Npn \__enumext_standard_item_viii:w
3390    {
3391      \bool_set_false:N \l__enumext_item_starred_viii_bool
3392        \peek_meaning:NTF [
3393          {
3394            \bool_set_eq:NN
3395              \l__enumext_wrap_label_viii_bool
3396              \l__enumext_wrap_label_opt_viii_bool
3397            \__enumext_start_item_viii:w
3398          }
3399          {
3400            \bool_set_true:N \l__enumext_wrap_label_viii_bool
3401            \legacy_if_set_true:n { @noitemarg }
3402            \__enumext_start_item_viii:w [ \l__enumext_label_viii_tl ]
3403          }
3404    }
```

(*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*⟩].

```
3405  \cs_new_protected:Npn \__enumext_starred_item_viii:w
3406    {
3407      \bool_set_true:N \l__enumext_item_starred_viii_bool
3408      \bool_set_true:N \l__enumext_wrap_label_viii_bool
3409      \peek_meaning:NTF [
3410        { \__enumext_starred_item_viii_aux_i:w }
3411        { \__enumext_starred_item_viii_aux_ii:w }
3412    }
```

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*⟩.

```
3413  \cs_new_protected:Npn \__enumext_starred_item_viii_aux_i:w [#1]
3414    {
```

```
3415        \tl_clear:N \l__enumext_store_keyans_label_tl
3416        \tl_if_novalue:nF { #1 }
3417          {
3418            \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_tl
3419              {
3420                \tl_put_right:Ne \l__enumext_store_keyans_label_tl { \l__enumext_store_keyans_item_op
3421                \tl_put_right:Ne \l__enumext_store_keyans_label_tl { #1 }
3422              }
3423            \tl_set:Ne \l__enumext_keyans_item_opt_tl { #1 }
3424          }
3425        \__enumext_starred_item_viii_aux_ii:w
3426      }
3427  \cs_new_protected:Npn \__enumext_starred_item_viii_aux_ii:w
3428    {
3429      \legacy_if_set_true:n { @noitemarg }
3430      \__enumext_start_item_viii:w [ \l__enumext_label_viii_tl ]
3431    }
```

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

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

```
3432  \cs_new_protected:Nn \__enumext_starred_item_exec:
3433    {
3434      \tl_put_left:Ne \l__enumext_store_keyans_label_tl { \l__enumext_label_viii_tl }
3435      \__enumext_store_addto_prop:V \l__enumext_store_keyans_label_tl
3436      \__enumext_keyans_store_ref:
3437      \tl_put_left:Ne \l__enumext_store_keyans_label_tl { \item }
3438      \__enumext_keyans_addto_seq_link:
3439      \bool_if:NT \l__enumext_show_answer_bool
3440        {
3441          \__enumext_print_keyans_box:NN \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
3442        }
3443      \bool_if:NT \l__enumext_show_position_bool
3444        {
3445          \tl_set:Ne \l__enumext_mark_answer_sym_tl
3446            {
3447              \group_begin:
3448                \exp_not:N \normalfont
3449                \exp_not:N \footnotesize [ \int_eval:n
3450                  {
3451                    \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
3452                  }
3453                ]
3454              \group_end:
3455            }
3456          \__enumext_print_keyans_box:NN \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
3457        }
3458    }
```

(*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

```
3459  \cs_new_protected_nopar:Npn \__enumext_start_item_viii:w [#1]
3460    {
3461      \cs_set_eq:NN \__enumext_stop_item_tmp_viii: \__enumext_stop_item_viii:
3462      \legacy_if:nT { @noitemarg }
3463        {
3464          \legacy_if_set_false:n { @noitemarg }
3465          \legacy_if:nT { @nmbrlist }
3466            {
3467              \bool_if:NT \l__enumext_hyperref_bool
3468                {
3469                  \legacy_if_set_true:n { @hyper@item }
3470                }
```

```
3471            \refstepcounter{enumXviii}
3472          }
3473        }
```

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

```
3474      \group_begin:
3475        \lrbox{ \l__enumext_item_text_viii_box }
3476          \bool_if:NF \l__enumext_footnotes_key_bool
3477            {
3478              \__enumext_renew_footnote:
3479            }
3480          \bool_if:NT \l__enumext_item_starred_viii_bool
3481            {
3482              \__enumext_starred_item_exec:
3483            }
3484          \group_begin:
3485            \tl_use:N \l__enumext_label_font_style_viii_tl
3486            \bool_if:NTF \l__enumext_wrap_label_viii_bool
3487              {
3488                \makebox[ \l__enumext_labelwidth_viii_dim ][ \l__enumext_align_label_viii_str ]
3489                  { \__enumext_wrapper_label_viii:n {#1} }
3490              }
3491              {
3492                \makebox[ \l__enumext_labelwidth_viii_dim ][ \l__enumext_align_label_viii_str ]{ #1
3493              }
3494          \group_end:
3495          \skip_horizontal:N \l__enumext_labelsep_viii_dim
3496          \tl_use:N \l__enumext_after_list_args_viii_tl
3497          \__enumext_minipage:w [ t ]{ \l__enumext_joined_width_viii_dim  }
3498            \skip_set_eq:NN \parindent \l__enumext_listparindent_viii_dim
3499            \skip_set_eq:NN \parskip \l__enumext_parsep_viii_skip
3500            \bool_if:NT \l__enumext_item_starred_viii_bool
3501              {
3502                \tl_use:N \l__enumext_fake_item_indent_viii_tl
3503                \__enumext_keyans_show_item_opt: \skip_horizontal:n { -\l__enumext_fake_item_indent_
3504              }
3505              {
3506                \tl_use:N \l__enumext_fake_item_indent_viii_tl
3507              }
3508        }
```

(*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.

```
3509    \cs_new_protected_nopar:Nn \__enumext_stop_item_viii:
3510      {
3511          \__enumext_endminipage:
3512        \endlrbox
3513      \group_end:
3514      \box_set_wd:Nn \l__enumext_item_text_viii_box
3515        {
3516          \l__enumext_joined_width_viii_dim
3517          + \l__enumext_labelwidth_viii_dim
3518          + \l__enumext_labelsep_viii_dim
3519        }
3520      \int_set:Nn \hbadness { 10000 }
3521      \box_use:N \l__enumext_item_text_viii_box
3522      \bool_if:NF \l__enumext_footnotes_key_bool
3523        {
3524          \__enumext_print_footnote:
3525        }
3526      \int_compare:nNnTF { \l__enumext_item_column_pos_viii_int } = { \l__enumext_columns_viii_int 
3527        {
3528          \par\noindent
3529          \int_zero:N \l__enumext_item_column_pos_viii_int
3530        }
3531        { \hspace{ \l__enumext_columns_sep_viii_dim } }
3532    }
```

(*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.

```
3533 \cs_new_protected:Nn \__enumext_remove_extra_parsep_viii:
3534   {
3535     \int_compare:nNnT
3536       {
3537         \int_mod:nn {  \g__enumext_item_count_all_viii_int } { \l__enumext_columns_viii_int }
3538       }
3539       =
3540       { \c_zero_int }
3541       {
3542         \par
3543         \vspace{ -\l__enumext_itemsep_viii_skip }
3544         \int_gzero:N \g__enumext_item_count_all_viii_int
3545       }
3546   }
```

(*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.

```
3547 \NewDocumentCommand \getkeyans { m }
3548   {
3549     \exp_args:Ne \__enumext_getkeyans_aux:n
3550       { \tl_to_str:e { \text_expand:n {#1} } }
3551   }
```

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

__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.

```
3552 \cs_new_protected:Npn \__enumext_getkeyans_aux:n #1
3553   {
3554     \str_if_in:nnTF {#1} { : }
3555       {
3556         \use:e
3557           {
3558             \cs_set:Npn \exp_not:N \__enumext_tmp:w ##1 \c_colon_str ##2 \scan_stop:
3559               { {##1} {##2} }
3560           }
3561         \exp_after:wN \__enumext_getkeyans:nn \__enumext_tmp:w #1 \scan_stop:
3562       }
3563       { \msg_error:nnn { enumext } { missing-colon } {#1} }
3564   }
```

(*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*⟩.

```
3565 \cs_new_protected:Npn \__enumext_getkeyans:nn #1 #2
3566   {
3567     \prop_if_exist:cF { g__enumext_#1_prop }
3568       { \msg_error:nnn { enumext } { undefined-storage-anskey } {#1} }
3569     \group_begin:
3570       \prop_item:cn { g__enumext_#1_prop }{#2}
3571     \group_end:
3572   }
```

(*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.

```
3573 \keys_define:nn  { keyanskey / print }
3574   {
3575     level-1 .code:n     = \tl_put_right:Nn \l__enumext_print_keyans_i_tl
3576                           {
3577                             \setenumext[level,1] {#1} \setenumext[print,1] {#1}
3578                           },
3579     level-1 .initial:n = { label=\arabic*., nosep, columns=2, first=\small, font=\small },
3580     level-2 .code:n     = \tl_put_right:Nn \l__enumext_print_keyans_ii_tl
3581                           {
3582                             \setenumext[level,2] {#1} \setenumext[print,2] {#1}
3583                           },
3584     level-2 .initial:n = { nosep, label=(\alph*), first=\small, font=\small },
3585     level-3 .code:n     = \tl_put_right:Nn \l__enumext_print_keyans_iii_tl
3586                           {
3587                             \setenumext[level,3] {#1} \setenumext[print,3] {#1}
3588                           },
3589     level-3 .initial:n = { nosep, label=\roman*., first=\small, font=\small },
3590     level-4 .code:n     = \tl_put_right:Nn \l__enumext_print_keyans_iv_tl
3591                           {
3592                             \setenumext[level,4] {#1} \setenumext[print,4] {#1}
3593                           },
3594     level-4 .initial:n = { nosep, label=\Alph*., first=\small, font=\small },
3595     level-* .code:n     = \tl_put_right:Nn \l__enumext_print_keyans_vii_tl % starred
3596                           {
3597                             \setenumext[enumext*] {#1} %%\setenumext[print,*] {#1}
3598                           },
3599     level-* .initial:n = { label=\arabic*., nosep, columns=2, first=\small, font=\small },
3600   }
```

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

```
3601 \NewDocumentCommand \printkeyans { s O{} m }
3602   {
3603     \group_begin:
3604       \tl_use:N \l__enumext_print_keyans_i_tl
3605       \tl_use:N \l__enumext_print_keyans_ii_tl
3606       \tl_use:N \l__enumext_print_keyans_iii_tl
3607       \tl_use:N \l__enumext_print_keyans_iv_tl
3608       \tl_use:N \l__enumext_print_keyans_vii_tl
3609       \__enumext_printkeyans:nnn { #1 } { #2 } { #3 }
3610     \group_end:
3611   }
```

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

__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

```
3612 \cs_new_protected:Npn \__enumext_printkeyans:nnn #1 #2 #3
3613   {
3614     \seq_if_exist:cTF { g__enumext_#3_seq }
3615       {
3616         \seq_if_empty:cF { g__enumext_#3_seq }
3617           {
3618             %%\seq_show:c { g__enumext_#3_seq }
3619             \bool_if:nTF {#1}
3620               {
3621                 \begin{enumext*}[#2]
3622                   \seq_map_inline:cn { g__enumext_#3_seq } { ##1 }
3623                 \end{enumext*}
3624               }
3625               {
```

```
3626                  \begin{enumext}[#2]
3627                    \seq_map_inline:cn { g__enumext_#3_seq } { ##1 }
3628                  \end{enumext}
3629                }
3630              }
3631          }
3632          {
3633            \msg_error:nnn { enumext } { undefined-storage-anskey } {#3}
3634          }
3635      }
```

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

10.39 The command \setenumext

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

```
3636  \keys_define:nn { enumext / meta-families }
3637    {
3638      level-1  .code:n = { \keys_set:nn { enumext / level-1  } {#1} } ,
3639      level-2  .code:n = { \keys_set:nn { enumext / level-2  } {#1} } ,
3640      level-3  .code:n = { \keys_set:nn { enumext / level-3  } {#1} } ,
3641      level-4  .code:n = { \keys_set:nn { enumext / level-4  } {#1} } ,
3642      keyans   .code:n = { \keys_set:nn { enumext / keyans   } {#1} } ,
3643      enumext* .code:n = { \keys_set:nn { enumext / enumext* } {#1} } ,
3644      keyans*  .code:n = { \keys_set:nn { enumext / keyans*  } {#1} } ,
3645      print-1  .code:n = { \keys_set:nn { keyanskey / print } { level-1 = {#1} } } ,
3646      print-2  .code:n = { \keys_set:nn { keyanskey / print } { level-2 = {#1} } } ,
3647      print-3  .code:n = { \keys_set:nn { keyanskey / print } { level-3 = {#1} } } ,
3648      print-4  .code:n = { \keys_set:nn { keyanskey / print } { level-4 = {#1} } } ,
3649      print-*  .code:n = { \keys_set:nn { keyanskey / print } { level-* = {#1} } } ,
3650      unknown  .code:n = { \msg_error:nn { enumext } { unknown-key-family } } ,
3651    }
```

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

```
3652  \seq_const_from_clist:Nn \c__enumext_all_families_seq
3653    {
3654      level-1 , level-2 , level-3 , level-4 , keyans, enumext*,
3655      keyans* , print-1 , print-2 , print-3 , print-4 , print-*,
3656    }
```

\setenumext   Now we define the user command \setenumext.

```
3657  \NewDocumentCommand \setenumext { o +m }
3658    {
3659      \tl_if_novalue:nTF {#1}
3660        {
3661          \seq_map_inline:Nn \c__enumext_all_families_seq
3662        }
3663        {
3664          \seq_clear:N \l__enumext_setkey_tmpa_seq
3665          \seq_set_from_clist:Nn \l__enumext_setkey_tmpb_seq {#1}
3666          \int_set:Nn \l__enumext_setkey_tmpa_int
3667            {
3668              \seq_count:N \l__enumext_setkey_tmpb_seq
3669            }
3670          \int_compare:nNnTF { \l__enumext_setkey_tmpa_int } > { 1 }
3671            {
3672              \seq_pop_left:NN \l__enumext_setkey_tmpb_seq \l__enumext_setkey_tmpa_tl
3673              \seq_map_function:NN \l__enumext_setkey_tmpb_seq \__enumext_set_parse:n
3674              \seq_set_map_e:NNn \l__enumext_setkey_tmpa_seq \l__enumext_setkey_tmpa_seq
3675                {
3676                  \tl_use:N \l__enumext_setkey_tmpa_tl - ##1
3677                }
3678            }
3679            {
3680              \seq_put_right:Ne \l__enumext_setkey_tmpa_seq { \tl_trim_spaces:n {#1} }
3681            }
3682          \seq_if_empty:NTF \l__enumext_setkey_tmpa_seq
3683            { \seq_map_inline:Nn \c__enumext_all_families_seq }
3684            { \seq_map_inline:Nn \l__enumext_setkey_tmpa_seq }
3685        }
3686        {
3687          \keys_set:nn { enumext / meta-families } { ##1 = {#2} }
```

```
3688                }
3689            }
```

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

`\__enumext_set_parse:n`
`\__enumext_set_error:nn`

Internal functions used by the `\setenumext` command.

```
3690  \cs_new_protected:Npn \__enumext_set_parse:n #1
3691    {
3692      \tl_set:Ne \l__enumext_setkey_tmpb_tl { \tl_trim_spaces:n {#1} }
3693      \int_step_inline:nnn { 0 } { 4 } % <- max level
3694        { \tl_remove_all:Nn \l__enumext_setkey_tmpb_tl {##1} }
3695      \tl_if_empty:NTF \l__enumext_setkey_tmpb_tl
3696        {
3697          \seq_put_right:Ne \l__enumext_setkey_tmpa_seq
3698            { \tl_trim_spaces:n {#1} }
3699        }
3700        { \__enumext_set_error:nn {#1} { } }
3701    }
3702  \cs_new_protected:Npn \__enumext_set_error:nn #1 #2
3703    { \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.

```
3704  \msg_new:nnn { enumext } { package-load }
3705    {
3706      The ~ '#1' ~ package ~ is ~ already ~ loaded.
3707    }
3708  \msg_new:nnn { enumext } { package-not-load }
3709    {
3710      The ~ '#1' ~ package ~ will ~ be ~ loaded ~ as ~ a ~ dependency.
3711    }
3712  \msg_new:nnn { enumext } { package-load-foot }
3713    {
3714      The ~ '#1' ~ package ~ is ~ loaded ~ with ~ the ~ option ~ '#2'.
3715    }
```

Message used in the creation of counters by enumext package.

```
3716  \msg_new:nnn { enumext } { counters }
3717    {
3718      The ~ counter ~ '#1' ~ is ~ already ~ defined ~ by ~ some ~ \\
3719      package ~ or ~ macro, ~ it ~ cannot ~ be ~ continued.
3720    }
```

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

```
3721  \msg_new:nnn { enumext } { invalid-key }
3722    {
3723      The ~ key ~ '#1' ~ is ~ not ~ know ~ the ~ level ~ #2.
3724    }
3725  \msg_new:nnn { enumext } { unknown-key-family }
3726    {
3727      Unknown~key~family~`\l_keys_key_str'~for~enumext.
3728    }
```

Messages used in length calculation.

```
3729  \msg_new:nnn { enumext } { width-negative }
3730    {
3731      Ignoring ~ negative ~ value ~ '#1=#2' ~ \msg_line_context:.\\
3732      The ~ key ~ '#1'~ accepts ~ values  ~ >= ~ 0pt.
3733    }
3734  \msg_new:nnn { enumext } { width-zero }
3735    {
3736      Invalid ~ '#1=#2' ~ \msg_line_context:.\\
3737      The ~ key ~ '#1'~ accepts ~ values  ~ > ~ 0pt.
3738    }
```

Messages used by show-length key in enumext.

```
3739  \msg_new:nnn { enumext } { list-lengths }
3740    {
3741      **** ~ Lengths ~ used ~ by ~ 'enumext' ~ level ~ '#2' ~ \msg_line_context:~\c_space_tl ****\\
3742      \__enumext_show_length:nnn { dim  } { labelsep     } {#1}
3743      \__enumext_show_length:nnn { dim  } { labelwidth   } {#1}
3744      \__enumext_show_length:nnn { dim  } { itemindent   } {#1}
3745      \__enumext_show_length:nnn { dim  } { leftmargin   } {#1}
3746      \__enumext_show_length:nnn { dim  } { rightmargin  } {#1}
3747      \__enumext_show_length:nnn { dim  } { listparindent } {#1}
3748      \__enumext_show_length:nnn { skip } { topsep     } {#1}
3749      \__enumext_show_length:nnn { skip } { parsep     } {#1}
3750      \__enumext_show_length:nnn { skip } { partopsep } {#1}
3751      \__enumext_show_length:nnn { skip } { itemsep    } {#1}
3752      ****************************************************
3753    }
```

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

```
3754  \msg_new:nnn { enumext } { list-lengths-not-nested }
3755    {
3756      **** ~ Lengths ~ used ~ by ~ '#2' ~ environment ~ \msg_line_context:~\c_space_tl ****\\
3757      \__enumext_show_length:nnn { dim  } { labelsep     } {#1}
3758      \__enumext_show_length:nnn { dim  } { labelwidth   } {#1}
3759      \__enumext_show_length:nnn { dim  } { itemindent   } {#1}
3760      \__enumext_show_length:nnn { dim  } { leftmargin   } {#1}
3761      \__enumext_show_length:nnn { dim  } { rightmargin  } {#1}
3762      \__enumext_show_length:nnn { dim  } { listparindent } {#1}
3763      \__enumext_show_length:nnn { skip } { topsep     } {#1}
3764      \__enumext_show_length:nnn { skip } { parsep     } {#1}
3765      \__enumext_show_length:nnn { skip } { partopsep } {#1}
3766      \__enumext_show_length:nnn { skip } { itemsep    } {#1}
3767      ****************************************************
3768    }
```

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

```
3769  \msg_new:nnn { enumext } { items-same-answer }
3770    {
3771      **********~Checking~answers~on~'#1'~OK~**********\\
3772      **~ All ~ items ~ stored ~ in ~ sequence ~ '#1' ~ have ~ an ~ answer. \\
3773      ************************************
3774      \prg_replicate:nn { 7 + \str_count:n {#1} } { * }
3775    }
3776  \msg_new:nnn { enumext } { item-different-answer }
3777    {
3778      Number ~ of ~ items ~ different ~  of ~ number ~ of ~
3779      answer ~ in ~ sequence ~ '#1'~ closed ~ \msg_line_context:.
3780    }
```

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

```
3781  \msg_new:nnn { enumext } { missing-starred }
3782    {
3783      Missing ~ '\c_backslash_str #1*' ~ in ~ '#2' ~ \msg_line_context:.
3784    }
```

Message for the nesting depth of the environment enumext.

```
3785  \msg_new:nnn { enumext } { list-too-deep }
3786    {
3787      Too ~ deep ~ nesting  ~ for ~ 'enumext' ~ \msg_line_context:.~ \\
3788      The ~ maximum  ~ level  ~ of  ~ nesting  ~ is ~ 4.
3789    }
```

Messages used by \anskey and \anspic commands.

```
3790  \msg_new:nnn { enumext } { anskey-wrong-place }
3791    {
3792      Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:.~ \\
3793      '\c_backslash_str #1' ~ works ~ in ~ the ~ environment ~ '#2'.
3794    }
3795  \msg_new:nnn { enumext } { anspic-wrong-place }
3796    {
3797      Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:.~ \\
3798      '\c_backslash_str #1' ~ works ~ in ~ the ~ environment ~ '#2'.
3799    }
3800  \msg_new:nnn { enumext } { command-wrong-place }
```

```
3801    {
3802      Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:.~ \\
3803      '\c_backslash_str #1' ~ works ~ outside ~ the ~ environment ~ '#2'.
3804    }
```

Messages used by keyans and keyanspic environment.

```
3805  \msg_new:nnn { enumext } { keyans-nested }
3806    {
3807      The ~ environment ~ 'keyans' ~ can't ~ be  ~ nested  ~ \msg_line_context:.
3808    }
3809  \msg_new:nnn { enumext } { keyans-wrong-level }
3810    {
3811      Wrong ~ level ~ position ~ for ~ 'keyans' ~ \msg_line_context:.~ \\
3812      The ~ environment ~ 'keyans' ~ can ~ only ~ be ~ in ~ the ~ first ~ level.
3813    }
3814  \msg_new:nnn { enumext } { wrong-place }
3815    {
3816      Wrong ~ place ~ for ~ '#1' ~ environment ~\msg_line_context:.~ \\
3817      '#1' ~ is ~ only ~ found ~ with ~ '#2' ~  in  ~  'enumext.
3818    }
3819  \msg_new:nnn { enumext } { keyanspic-nested }
3820    {
3821      The ~ environment ~ 'keyanspic' ~ can't ~ be  ~ nested~ \msg_line_context:.~.
3822    }
3823  \msg_new:nnn { enumext } { keyanspic-wrong-level }
3824    {
3825      Wrong ~ level ~ position ~ for ~ 'keyanspic' ~ \msg_line_context:.~ \\
3826      The ~ environment ~ 'keyans' ~ can ~ only ~ be ~ in ~ the ~ first ~ level.
3827    }
```

Messages used by \getkeyans command.

```
3828  \msg_new:nnn { enumext } { undefined-storage-anskey }
3829    {
3830      Storage ~ named ~ '#1' ~ is ~ not ~ defined ~ \msg_line_context:.
3831    }
```

Messages used by \miniright command.

```
3832  \msg_new:nnn { enumext } { missing-miniright }
3833    {
3834      Missing ~ '\c_backslash_str miniright' ~ in ~ \msg_line_context:.\\
3835      The ~ key ~ 'mini-env' ~ need ~ '\c_backslash_str miniright'.
3836    }
3837  \msg_new:nnn { enumext } { wrong-miniright-place }
3838    {
3839      Wrong ~ place ~ for ~ '\c_backslash_str miniright' ~ \msg_line_context:.~ \\
3840      Works ~ in ~ 'enumext' ~ and ~ 'keyans' ~ with ~ key ~ 'mini-env'.
3841    }
3842  \msg_new:nnn { enumext } { wrong-miniright-use }
3843    {
3844      Wrong ~ use ~ for ~ '\c_backslash_str miniright' ~ \msg_line_context:.~ \\
3845      '\c_backslash_str miniright' ~ need ~ a ~ key ~ 'mini-env'.
3846    }
```

Messages used by enumext* and keyans* environments.

```
3847  \msg_new:nnn { enumext } { nested }
3848    {
3849      The ~ starred ~ environment ~ can't ~ be ~ nested ~ \msg_line_context:.
3850    }
3851  \msg_new:nnn { enumext } { item-joined }
3852    {
3853      Items ~ joined ~ (#1) ~ > ~ #2  ~ columns ~\msg_line_context:.
3854    }
3855  \msg_new:nnn { enumext } { item-joined-columns }
3856    {
3857      Not ~ space ~ to ~ join ~ items ~ (#1) ~ > ~ #2 ~\msg_line_context:.
3858    }
```

10.41   Finish package

Finish package implementation.

```
3859  \file_input_stop:
3860  ⟨/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.