enumext

ENUMERATE EXERCISE SHEETS

V1.0   2024-06-12[*]

©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 multicol and scontents packages 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 2024-06-12.
[†]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] and scontents[4] packages, 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. E), A duck *
* 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[6], 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[6] 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 of two main list environments `enumext` (vertical) and `enumext*` (horizontal), the environment `anskey*` and the command `\anskey` to *"store content"* and the environments `keyans`, `keyans*` and `keyanspic` for multiple choice. It also provides the commands `\getkeyans` to print 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 fatal 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.
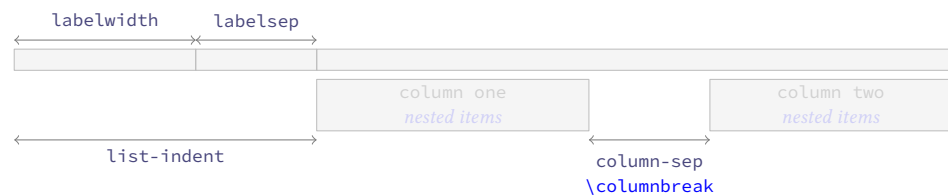

Figure 4: Representation of the two column output for a nested level in `enumext` environment.

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. It can be used in all nesting levels of the environment `enumext` and the environment `keyans` and can together with the `mini-env` key. If you need to force a start a new column `\columnbreak` must be used (see §4.5).

💣 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 on *"left side"* and *"right side"* is always used with *"aligned on top"* [t]. It can be used in all nesting levels of the environment `enumext` and the environment `keyans` 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 §4.6).

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

This package provides a user interface like the `enumitem`[6] package to customize the references which is activated by the `ref` key (§4.1), 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` (§5.1.1) when the key `save-ans` (§5.1) is active.

💣 The implementation of `\label` and `\ref` together with the `save-ref` key are compatible with the `hyperref`[8] package.

1.3.5 Support for `\footnote`

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

The best way to solve this is to use Jean-François Burnol `footnotehyper`[9] 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 environments provided

The package enumext provides two main list environments, the *vertical* environment enumext and the *horizontal* 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*}
```

2.1   The environment enumext

The enumext is an 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 with columns=2

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.

2.2   The environment enumext*

The enumext* is a *horizontal list environment* similar to the enumerate* environment provided by the enumitem package or task environment provided by the task package , \item and \item[⟨custom⟩] work as usual. The options can be configured globally using \setenumext command and locally using [⟨key = val⟩] in the environment.

Some considerations to take into account for this environment:

- The environment cannot be nested within itself, but it can be nested within enumext and can contain it nested within it.

- Each *"item"* in the environment is placed within a minipage environment whose *width* is stored in the dimension \itemwidth that includes labelwith, labelsep plus the *width of the content.*

- You cannot have floating environments like figure or table but \footnote with hyperref support is supported if the footnotehyper package is loaded.

Example with columns=2

1. This text is in the first level.

X This text is in the first level.

2. This text is in the first level.

⋆ 3. This text is in the first level.

2.3   The command \item*

\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 argument* '*' 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 and enumext* environments is NOT the same as in the keyans and keyans* environments.

2.3.1 Keys for \item*

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 and enumext*. The *symbol* can be in text or math mode, for example item-sym*={$\ast$}.

item-pos* = {⟨rigid length⟩} *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*.

2.4 The command \item in enumext*

The \item command for the enumext* environment provides an optional *"first argument"* \item(⟨columns⟩) which *"joins items"* between columns. Let's consider the following examples adapted directly from the task package:

```
\begin{enumext*}[widest=10,columns=4]
  \item The first
  \item* The second
  \item The third
  \item The fourth
  \item(3)* The fifth item is way too long for this and needs three columns
  \item The sixth
  \item the seventh
  \item(2)[X] The eighth item is way too long for this and needs two columns
  \item[Z] The ninth
  \item The tenth
\end{enumext*}
```

1. The first   ⋆ 2. The second   3. The third   4. The fourth

⋆ 5. The fifth item is way too long for this and needs three columns   6. The sixth

7. the seventh   X The eighth item is way too long for this and needs Z The ninth
two columns

8. The tenth

3 The command \setenumext

\setenumext

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

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

💣 The key save-ans that activate the *"storage system"* must NOT be passed through this command and must be passed directly in the optional argument of the *"first level"* of the environment in which they are executed.

4 The keyval system

The ⟨key = val⟩ system used by the enumext package is implemented using l3keys so it must be taken into consideration that those keys marked as *"value forbidden"*, that is ⟨key⟩ is different from ⟨key=⟩.

All ⟨keys⟩ described in this section are available for the enumext, enumext*, keyans and keyans* environments with the exception of the keys series, resume, resume* which are only available for the *"first level"* of the environments enumext and enumext*; and the keys mini-right, mini-right* which are only available for the enumext* and keyans* environments.

All ⟨keys⟩ related to vertical or horizontal spacing accept a *"skip"* or *"dim"* expression if passed between braces, i.e. you do not need to use \dimeval or \dimexpr to perform calculations.

It should be kept in mind that using any ⟨key⟩ that sets a *rubber lengths* or *rigid lengths* for vertical or horizontal space on a level will influence the vertical and horizontal space for *inners levels* and keyans, keyans* and keyanspic environments.

4.1 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 the first level of the environments `enumext` and `enumext*` are `\arabic*.`, for second level are (`\alph*`), for third level are `\roman*.` and for fourth level are `\Alph*.`. For `keyans` and `keyans*` environments the default value is `\Alph*)`.

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

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

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

Internally it renews the command associated with each counter when it is executed, i.e., in the environment `enumext` the command `\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*⟩`]`.

4.2 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`.

4.2.1 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 value for the first level of the environments `enumext` and `enumext*` 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`. For `keyans` and `keyans*` environments the default value is `4.0pt plus 2.0pt 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 value for the first level of the environments `enumext` and `enumext*` 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`. For `keyans` and `keyans*` environments the default value is `2.0pt plus 1.0pt minus 1.0pt`.

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 in environment `enumext` are `2.0pt plus 1.0pt minus 1.0pt`, for third and fourth level are `1.0pt minus 1.0pt`. For `keyans`, `keyans*` and `enumext*` environments the default value is `2.0pt plus 1.0pt minus 1.0pt`.

💣 The value of this parameter also affects the *inner levels* and the environments `keyans`, `keyanspic` and `keyans*`. 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 value for the first level of the environments `enumext` and `enumext*` are `4.0pt plus 2.0pt minus 1.0pt`, for the rest of the levels are `2.0pt plus 1.0pt minus 1.0pt`. For `keyans` and `keyans*` environments the default value is `4.0pt plus 2.0pt 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.

base-fix ⟨*value forbidden*⟩   default: *not used*

This is a *"meta-key"* that does not receive an argument available only for the *first level* of environment `enumext` and environment `enumext*`. Fix the *baseline* when an environment `enumext` is nested in `enumext*` or vice versa and there is no material between the `\item` and the start of the environment for example `\item \begin{enumext*}` within the environment `enumext`. Internally sets the keys `topsep`, `above` and `above*` at `0pt`.

💣 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"*.

4.2.2 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` is set in the environment `enumext` 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.

💣 For the `enumext*` and `keyans*` environments the keys `list-indent` and `list-offset` have the same effect.

4.3 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.

4.4 Keys for start, series and resume

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

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

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

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

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

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

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

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

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

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

4.5 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.

4.6 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`.

4.6.1 The command `\miniright`

`\miniright` `\begin{enumext}[mini-env=`⟨*rigid length*⟩`]` ⟨*item's before*⟩ `\item \miniright` ⟨*content*⟩ `\end{enumext}`
`\begin{enumext}[mini-env=`⟨*rigid length*⟩`]` ⟨*item's before*⟩ `\item \miniright*`⟨*content*⟩ `\end{enumext}`

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 argument* '`*`' 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 (see §1.3.5 for full support).

4.6.2 The key `mini-right`

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

mini-right = {⟨*content*⟩}   default: *not used*

Set the *content* for the drawing or tabular to be placed in the `minipage` environment on the *"right side"* by starting it with `\centering`. The `{`⟨*content*⟩`}` must be passed between braces.

mini-right* = {⟨*code for drawing or tabular*⟩}   default: *not used*

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

💣 The keys `mini-right` and `mini-right*` has a *slightly different* implementation, the argument `{`⟨*content*⟩`}` is saved in a box and then printed outside the environment using *hooks*.

5   The storage system

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

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

By executing the key `save-ans={`⟨*store name*⟩`}` the entire structure of the environment (excluding the first level) including the optional arguments passed to the inner levels or the environment nested in it, along with the content passed to `\anskey`, the current ⟨*labels*⟩ for `\item*` and `\anspic*` in the environments `keyans`, `keyans*` and `keyanspic` will be stored in a ⟨*sequence*⟩ and at the same time will be stored (without the environment structure or optional arguments) in a ⟨*prop list*⟩.

The optional arguments of the inner levels or the nested environment are filtered by excluding all ⟨*keys*⟩ related to the *"stored system"* along with the keys `series`, `resume` and `resume*` when storing in ⟨*sequence*⟩.

5.1   Keys for storage system

🔖* The only ⟨*keys*⟩ available for all levels of the `enumext` environment and the `enumext*` environment are `no-store` and `save-key`, the rest of the ⟨*keys*⟩ described in this section must be passed directly in the optional argument of the *"first level"* of the environment in which the key `save-ans` is executed. The key `save-ans` should NOT be passed with the command `\setenumext`.

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

`save-key` = {⟨*key list*⟩}   default: *not set*

This key *overrides* the default *"stored keys"* of the optional arguments of the inner levels or nested environment that will be passed to the ⟨*sequence*⟩. The ⟨*key list*⟩ passed to this key ignores any ⟨*keys*⟩ in the *"stored system"* and must be passed between braces. For example, if we execute at a second level:

```
\begin{enumext}[save-ans={⟨store name⟩}]
  \item Text \anskey{answer}
  \item Text
    \begin{enumext}[nosep, columns=2, save-key={columns=3}]
          ...
    \end{enumext}
\end{enumext}
```

The ⟨*keys*⟩ that will be stored by default in the ⟨*sequence*⟩ would be `nosep, columns=2`, but using the key `save-key={columns=3}` will overwrite this and store it in the ⟨*sequence*⟩ only the key `columns=3` ignoring all the others.

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

Sets the *text symbol* that will separate the current ⟨*label*⟩ to the *optional argument* passed to the `\item*` and `\anspic*` in the `keyans`, `keyans*` and `keyanspic` environments and storing them in the ⟨*store name*⟩ defined by the `save-ans` key. 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.

5.1.1   Keys for `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.

5.1.2 Keys for `wrap` and `display`

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

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

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

Displays the *argument* passed to the `\anskey`, the *body* for `anskey*` environment, the ⟨*label*⟩ for `\item*` and `\anspic*` at the place where it is executed. If the optional argument is present in `\item*` or `\anspic*` it will be shown using `wrap-opt` key.

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

Sets the *symbol* to be displayed in the left margin for `\anskey`, `anskey*`, `\item*` and `\anspic*` in the place where they are executed when using the key `show-ans`.

`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.

5.1.3 Keys for debug and checking

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

Displays the *position* occupied by the *"stored content"* by `\anskey`, `anskey*`, `\item*` and `\anspic*` in the *prop list* {⟨*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 displaying an appropriate message on the terminal. This key works under the logic that each `\item` or `\item*` that does not open an inner level or nested environment contains *"only one answer"* or *"only one execution"* of the `\anskey` or `anskey*`. It is intended to be used in conjunction with the `no-store` key.

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

This is a *meta-key* that does not receive an argument and disables the structure stored in the *sequence* {⟨*store name*⟩} set by `save-ans` key at the entire level or a nested environment in which it runs. This key is intended for use in internal levels or nested `enumext` or `enumext*` environments in which you want to use `enumext` or `enumext*` but *"without"* using the `\anskey`, *"without"* use `anskey*`, *"without"* interfering with the `check-ans` key and *"without"* storing an unwanted structure in the *sequence* {⟨*store name*⟩}.

5.2 The command `\anskey`

```
\anskey
```

`\anskey[`⟨*keys*⟩`]{`⟨*content*⟩`}`

The command `\anskey` takes a mandatory non empty argument {⟨*content*⟩} and *"stores"* it in the *sequence* and *prop list* {⟨*store name*⟩} set by `save-ans` key. By design the command cannot be nested or passed *verbatim material* in the argument and it is assumed that each *numbered* `\item` or `\item*` within the environment in which it is active it has a *"single execution"* of `\anskey` unless `\item` or `\item*` open a nested level or use the `no-store` key.

If `save-ref` key are active and the `hyperref`[8] package is detected, `\hyperlink` and `\hypertarget` will be used, otherwise the usual *"label and ref"* system provided by LaTeX will be used.

The `\anskey` command is available for all levels of the `enumext` environment and the `enumext*` environment, but is disabled for the `keyans`, `keyans*` and `keyanspic` environments.

5.2.1 Keys for `\anskey`

By default the {⟨*content*⟩} passed to `\anskey` when *"storing"* in the *sequence* {⟨*store name*⟩} has the form `\item` ⟨*content*⟩, the following ⟨*keys*⟩ allow modifying the way in which it is *"stored"* in the *sequence*.

`break-col` ⟨*value forbidden*⟩ default: *not used*

Stores {⟨*content*⟩} in the *sequence* {⟨*store name*⟩} of the form `\columnbreak \item` ⟨*content*⟩.

`item-join` = {⟨*columns*⟩} default: *not set*

Set the *number of columns* to be used for `\item(`⟨*columns*⟩`)` and stores {⟨*content*⟩} in the *sequence* {⟨*store name*⟩} of the form `\item(`⟨*columns*⟩`)` ⟨*content*⟩.

`item-star` ⟨*value forbidden*⟩ default: *not used*

Stores {⟨*content*⟩} in the *sequence* {⟨*store name*⟩} of the form `\item*` ⟨*content*⟩.

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

Sets the *symbol* for `\item*` when using the key `item-star` and stores {⟨*content*⟩} in the *sequence* {⟨*store name*⟩} of the form `\item*[`⟨*symbol*⟩`] `⟨*content*⟩. The *symbol* can be in text or math mode, for example `item-sym*={$\ast$}` stores `\item*[$\ast$] `⟨*content*⟩.

item-pos* = {⟨*rigid length*⟩} default: *not set*

Sets the *offset* for `\item*` when using the keys `item-star` and `item-sym*` and stores {⟨*content*⟩} in the *sequence* {⟨*store name*⟩} of the form `\item*[`⟨*symbol*⟩`][`⟨*offset*⟩`] `⟨*content*⟩.

Example

```
\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}
```

⋆ 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

5.3 The environment anskey*

anskey* `\begin{anskey*}[`⟨*key* = *val*⟩`] `⟨*body content*⟩` \end{anskey*}`

The environment `anskey*` takes a mandatory {⟨*body content*⟩} and *"stores"* it in the *sequence* and *prop list* {⟨*store name*⟩} set by `save-ans` key. If `save-ref` key are active and the `hyperref`[8] package is detected, `\hyperlink` and `\hypertarget` will be used, otherwise the usual *"label and ref"* system provided by LATEX will be used.

By design the environment cannot be nested but full supports *"verbatim material"* in the body and it is assumed that each numbered `\item` or `\item*` within the environment in which it is active it has a *"single execution"* unless `\item` or `\item*` open a nested level or use the `no-store` key.

The `anskey*` environment is implemented using the `scontents` package, for the correct operation `\begin{anskey*}` and `\end{anskey*}` must be in different lines, all ⟨*keys*⟩ must be passed separated by commas and "without separation" of the start of the environment. Comments "%" or "any character" after `\begin{anskey*}` or `[`⟨*key* = *val*⟩`]` on the same line are NOT supported, the package `scontents` will return an "error" message if this happens. In a similar way comments "%" or "any character" after `\end{anskey*}` on the same line the package `scontents` will return a "warning" message.

The `anskey*` environment uses the same ⟨*keys*⟩ as the `\anskey` command next to the keys `write-env`, `force-eol` and `overwrite` inherited from package `scontents`. The environment and is available for all levels of the `enumext` environment and the `enumext*` environment, but it is disabled for the `keyans`, `keyans*` and `keyanspic` environments.

💣 For security reasons the keys `store-env`, `print-env` and `write-out` they have been left disabled. It is recommended that you review the `scontents`[4] documentation to understand how the keys described here work.

Example

```
\begin{enumext}[save-ans=test,show-pos=true,start=5]
  \item* Text containing our instructions or questions.
    \begin{anskey*}[item-star]
      ⟨first answer⟩
    \end{anskey*}
  \item Text containing our instructions or questions.
    \begin{enumext}
      \item Question.
        \begin{anskey*}
          ⟨second answer⟩
        \end{anskey*}
    \end{enumext}
  \item Text containing our instructions or questions.
    \begin{anskey*}
      ⟨third answer⟩
    \end{anskey*}
  \item Text containing our instructions or questions.
```

```
      \begin{anskey*}
        ⟨fourth answer⟩
      \end{anskey*}
  \end{enumext}
```

★ 5. Text containing our instructions or questions.   7. Text containing our instructions or questions.

    [5] First answer with `verbatim`   [7] third answer

  6. Text containing our instructions or questions.   8. Text containing our instructions or questions.

    (a) Question.   [8] fourth answer

      [6] second answer

5.4 The environments keyans and 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` and `keyans*` environments are *"enumerated list"* environments designed for *"multiple choice"* questions activated by the `save-ans` key. This environments 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 and the command `\item(`⟨*columns*⟩`)` is available for the `keyans*` environment.

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

The ⟨*keys*⟩ set in the optional argument of the environment are the same (almost) as those of the `enumext` and `enumext*` environments and have higher precedence than those set by `\setenumext[`⟨*keyans*⟩`]{`⟨*key* = *val*⟩`}` or `\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*)`.

5.4.1 The \item* in keyans and 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 *sequence* and *prop list* {⟨*store name*⟩} set by `save-ans` key in the *"first level"* of the `enumext` or `enumext*` environments.

The *starred argument* '`*`' 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 `\item*` will only appear *"once"* within the environment.

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

Example

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

2. Text containing a question and image.
   A) Choice
   B) Choice
   C) Choice
   D) Choice
   * E) [note] Correct choice

Some text

5.5 The environment keyanspic

keyanspic `\begin{keyanspic}[⟨nº above, nº below⟩]\anspic{⟨drawing⟩}\anspic*[⟨content⟩]{⟨drawing⟩}`

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

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

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

5.5.1 The command `\anspic`

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

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

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

Example

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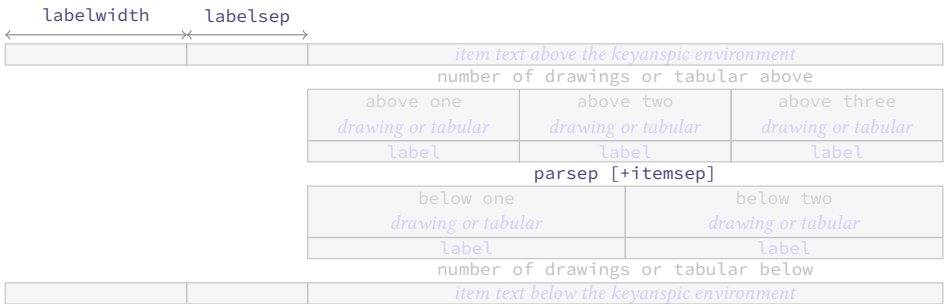
1. Question with images.

A)

B)

C)

D)

* E)[note]

5.6 Printing stored content

5.6.1 The command \getkeyans

\getkeyans

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

The command **\getkeyans** prints the *"stored content"* in *prop list* {⟨*store name*⟩} defined by `save-ans` key in the ⟨*position*⟩ returned by the `show-pos` key. The *"stored content"* can only be accessed *after* it is stored, if {⟨*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.

5.6.2 The command \printkeyans

\printkeyans

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

The command **\printkeyans** prints *"all stored content"* in *sequence* {⟨*store name*⟩} defined by `save-ans` key placing this inside the `enumext` environment or the `enumext*` environment if the *starred argument* '`*`' is used. The *"stored content"* can only be accessed *after* it is stored in the *sequence*, if {⟨*store name*⟩} does not exist the command will return an error.

The optional argument allows managing the ⟨*keys*⟩ in the *"first level"* of the environment in which the *"stored content"* of the *sequence* {⟨*store name*⟩} will be printed, if the *starred argument* '`*`' is used it will be `enumext*` otherwise `enumext`.

The default values for the *"first level"* are the same as the default values for the `enumext` and `enumext*` environments along with the keys `nosep,first=\small,font=\small` and `columns=2`. For the inner levels of the environment `enumext` saved in the *sequence* {⟨*store name*⟩} the default values are the same as those established for the second, third and fourth levels plus the keys `nosep,first=\small,font=\small`. If the environment `enumext*` is saved within the *sequence* {⟨*store name*⟩} it will have the same default values plus the keys `nosep,first=\small, font=\small`.

Since the command encapsulates by default the `enumext` environment or the `enumext*` environment, we must take some considerations:

- If we execute \printkeyans*{⟨*store name*⟩} and the *sequence* {⟨*store name*⟩} already contains any `enumext*` environment an error will be returned as we cannot nest.

- If we execute \printkeyans*{⟨*store name*⟩} and the *sequence* {⟨*store name*⟩} contains any `enumext` environments, they will start with the ⟨*keys*⟩ set for the first level unless they are set in the optional argument or `save-key` is used to modify it.

- If we execute \printkeyans{⟨*store name*⟩} and the *sequence* {⟨*store name*⟩} contains any environment `enumext*`, they will start with the ⟨*keys*⟩ set by default unless they are set in the optional argument or `save-key` is used to modify it.

The default values for the *"first level"* of \printkeyans commands and \printkeyans* are established using \setenumext[⟨*print* , *1*⟩]{⟨*keys*⟩} and \setenumext[⟨*print**⟩]{⟨*keys*⟩}. If we need to set the ⟨*keys*⟩ for the environment `enumext` *"saved"* in the *sequence* {⟨*store name*⟩} we will use \setenumext[⟨*print* , *level*⟩]{⟨*keys*⟩} and if we need to set the ⟨*keys*⟩ for the environment `enumext*` *"saved"* in the *sequence* {⟨*store name*⟩} we will use \setenumext[⟨*print* , *⟩]{⟨*keys*⟩}.

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] $\boxed{3(x + y + z)}$

2. True False
   (a) LaTeX2e is cool?

   [2] $\boxed{\text{Very True!}}$

3. Related to Linux
   (a) You use linux?

[3] $\boxed{\text{Yes}}$

(b) Rate the following package and class
   i. `xsim`

   [4] $\boxed{\text{very good}}$

   ii. `exsheets`

   [5] $\boxed{\text{obsolete}}$

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   *

6 Full examples

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

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

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

Example 1

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

1. La velocità di $1,00 \times 10^2$ m/s espressa in km/h è:
   - A 36 km/h.
   - B 360 km/h.
   - C 27,8 km/h.
   - D $3,60 \times 10^8$ km/h.

2. In fisica nucleare si usa l'angstrom (simbolo: $1\,\text{Å} = 1 \times 10^{-10}$ m) e il fermi o femtometro ($1\,\text{fm} = 1 \times 10^{-15}$ m). Qual è la relazione tra queste due unità di misura?
   - A $1\,\text{Å} = 1 \times 10^5$ fm.
   - B $1\,\text{Å} = 1 \times 10^{-5}$ fm.
   - C $1\,\text{Å} = 1 \times 10^{-15}$ fm.
   - D $1\,\text{Å} = 1 \times 10^3$ fm.

3. La velocità di $1,00 \times 10^2$ m/s espressa in km/h è:
   - A 36 km/h.
   - B 360 km/h.
   - C 27,8 km/h.
   - D $3,60 \times 10^8$ km/h.

4. In fisica nucleare si usa l'angstrom (simbolo: $1\,\text{Å} = 1 \times 10^{-10}$ m) e il fermi o femtometro ($1\,\text{fm} = 1 \times 10^{-15}$ m). Qual è la relazione tra queste due unità di misura?
   - A $1\,\text{Å} = 1 \times 10^5$ fm.
   - B $1\,\text{Å} = 1 \times 10^{-5}$ fm.
   - C $1\,\text{Å} = 1 \times 10^{-15}$ fm.
   - D $1\,\text{Å} = 1 \times 10^3$ fm.

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

Example 2

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

1. La velocità di $1,00 \times 10^2$ m/s espressa in km/h è:
   - A 36 km/h.
   - ✓ B 360 km/h.
   - C 27,8 km/h.
   - D $3,60 \times 10^8$ km/h.

2. In fisica nucleare si usa l'angstrom (simbolo: $1\,\text{Å} = 1 \times 10^{-10}$ m) e il fermi o femtometro ($1\,\text{fm} = 1 \times 10^{-15}$ m). Qual è la relazione tra queste due unità di misura?
   - ✓ A $1\,\text{Å} = 1 \times 10^5$ fm.
   - B $1\,\text{Å} = 1 \times 10^{-5}$ fm.
   - C $1\,\text{Å} = 1 \times 10^{-15}$ fm.

---

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

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 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   Ⓓ I and III only

   Ⓑ II only   Ⓔ I, II, and III

   Ⓒ I and II only

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

Example 4

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

🦆 Factor $x^2 - 2x + 1$

🦆 Factor $3x + 3y + 3z$

The following questions need to be cuaqtified :)

True False

(a) $\alpha > \delta$
(b) LaTeX2e is cool?

Related to Linux

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

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

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

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

Example 5

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

1

Which choice best describes what happens in the passage?

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

3

Which choice best describes what happens in the passage?

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

2

Which choice best describes what happens in the passage?

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

4

Which choice best describes what happens in the passage?

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

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

7   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[5] 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[11] and l3seq[11] modules together with the hyperref[8] and enumitem[6] 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 typesseting 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.

8   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] González, Pablo. "scontents - Stores LaTeX contents in memory or files". Available from CTAN, `https://www.ctan.org/pkg/scontents`, 2022.

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

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

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

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

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

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

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

[12] The LaTeX Project. "The LaTeX 2_ε sources". Available from CTAN, `https://ctan.org/tex-archive/macros/latex/base`, 2024.

[13] The LaTeX Project. "LaTeX for authors current version". Available from CTAN, `https://ctan.org/pkg/latex-base`, 2024.

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

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

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

9 Change history

v1.0 2024-06-12 – First public release.

10 Index of Documentation

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

11   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.

11.1   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.

11.2   Initial set up

Start the DocStrip guards.

```
1  ⟨*package⟩
```

Identify the internal prefix (LATEX3 DocStrip convention) for l3doc class.

```
2  ⟨@@=enumext⟩
```

11.3   Declaration of the package

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

```
3  \NeedsTeXFormat{LaTeX2e}[2024-06-01]
```

Now declare the enumext package.

```
4  \ProvidesExplPackage
5    {enumext}
6    {2024-06-12}
7    {1.0}
8    {Enumerate exercise sheets}
```

Finally check if the multicol and scontents packages are 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}[2024-05-23]
18       }
19     \IfPackageLoadedTF { scontents }
20       {
21         \msg_info:nnn { enumext } { package-load } { scontents }
22       }
23       {
24         \msg_info:nnn { enumext } { package-not-load } { scontents }
25         \RequirePackage{scontents}
26       }
27   }
```

11.4 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_anskey_level_int
\l__enumext_keyans_level_int
\l__enumext_keyans_level_h_int
\l__enumext_keyans_pic_level_int

Integer variables will control the nesting levels of the environments and `\anskey` command.

```
28 \int_new:N  \l__enumext_level_int
29 \int_new:N  \l__enumext_level_h_int
30 \int_new:N  \l__enumext_anskey_level_int
31 \int_new:N  \l__enumext_keyans_level_int
32 \int_new:N  \l__enumext_keyans_level_h_int
33 \int_new:N  \l__enumext_keyans_pic_level_int
```

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

\l__enumext_starred_bool
\g__enumext_starred_bool
\l__enumext_starred_first_bool
\l__enumext_standar_bool
\g__enumext_standar_bool
\l__enumext_standar_first_bool
\l__enumext_anskey_env_bool
\l__enumext_keyans_env_bool
\g__enumext_start_line_tl
\g__enumext_envir_name_tl

Internal variables used by functions `\__enumext_is_not_nested:`, `\__enumext_is_on_first_-level:` and `\__enumext_keyans_start_line:` (§11.5.1).

```
34 \bool_new:N \l__enumext_starred_bool
35 \bool_new:N \g__enumext_starred_bool
36 \bool_new:N \l__enumext_starred_first_bool
37 \bool_new:N \l__enumext_standar_bool
38 \bool_new:N \g__enumext_standar_bool
39 \bool_new:N \l__enumext_standar_first_bool
40 \bool_new:N \l__enumext_anskey_env_bool
41 \bool_new:N \l__enumext_keyans_env_bool
42 \tl_new:N   \g__enumext_start_line_tl
43 \tl_new:N   \g__enumext_envir_name_tl
```

(*End of definition for* `\l__enumext_starred_bool` *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` (§11.9) and then modified by the function `\__enumext_label_style:Nnn` used by label key (§11.12).

```
44 \cs_set_protected:Npn \__enumext_tmp:n #1
45   {
46     \tl_new:c { l__enumext_counter_#1_tl }
47   }
48 \clist_map_inline:nn { i, ii, iii, iv, v, vi, vii, viii } { \__enumext_tmp:n {#1} }
```

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

\c__enumext_counter_style_tl
\l__enumext_ref_key_arg_tl
\l__enumext_ref_the_count_tl
\l__enumext_the_counter_X_tl
\l__enumext_renew_the_count_X_tl

Internal variables used by ref key (§11.12).

```
49 \tl_const:Nn \c__enumext_counter_style_tl
50   { { arabic } { roman } { Roman } { alph } { Alph } }
51 \tl_new:N \l__enumext_ref_key_arg_tl
52 \tl_new:N \l__enumext_ref_the_count_tl
53 \cs_set_protected:Npn \__enumext_tmp:n #1
54   {
55     \tl_new:c  { l__enumext_renew_the_count_#1_tl }
56     \tl_new:c  { l__enumext_the_counter_#1_tl }
57     \tl_set:ce { l__enumext_the_counter_#1_tl } { \exp_not:c { theenumX#1 } }
58   }
59 \clist_map_inline:nn { i, ii, iii, iv, v, vi, vii, viii } { \__enumext_tmp:n {#1} }
```

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

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

Internal variables used by resume, resume* and series keys (§11.23).

```
60 \int_new:N  \g__enumext_resume_int
61 \int_new:N  \g__enumext_resume_vii_int
62 \tl_new:N   \l__enumext_resume_name_tl
63 \bool_new:N \l__enumext_resume_active_bool
64 \tl_new:N   \g__enumext_standar_series_tl
65 \tl_new:N   \g__enumext_starred_series_tl
```

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

\l__enumext_current_widest_dim
\g__enumext_counter_styles_tl
\g__enumext_widest_label_tl
\l__enumext_label_width_by_box

The variable `\l__enumext_current_widest_dim` stores the current label width, the variable `\g__enumext_counter_styles_tl` stores the default ⟨*label style*⟩ and the variable `\g__enumext_widest_label_tl` the label width. These variables are used by widest (§11.13) and label (§11.11) keys.

```
66 \dim_new:N \l__enumext_current_widest_dim
67 \tl_new:N  \g__enumext_counter_styles_tl
68 \tl_new:N  \g__enumext_widest_label_tl
69 \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 (§11.16). The variables `\l__enumext_leftmargin_X_dim` and `\l__enumext_itemindent_X_dim` are used and set by the function `\__enumext_calc_hspace:NNNNNNNNNNNNN` (§11.34.1).

```
70 \cs_set_protected:Npn \__enumext_tmp:n #1
71   {
72     \bool_new:c { l__enumext_leftmargin_tmp_#1_bool }
73     \dim_new:c  { l__enumext_leftmargin_tmp_#1_dim }
74     \dim_new:c  { l__enumext_leftmargin_#1_dim     }
75     \dim_new:c  { l__enumext_itemindent_#1_dim     }
76   }
77 \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 §11.20).

```
78 \cs_set_protected:Npn \__enumext_tmp:n #1
79   {
80     \skip_new:c  { l__enumext_multicols_above_#1_skip }
81     \skip_new:c  { l__enumext_multicols_below_#1_skip }
82   }
83 \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 (§11.21.4) and the keys mini-right, mini-right*, mini-env and mini-sep (§11.19, §11.21).

```
84 \int_new:N  \g__enumext_minipage_stat_int
85 \skip_new:N \l__enumext_minipage_left_skip
86 \skip_new:N \l__enumext_minipage_right_skip
87 \skip_new:N \l__enumext_minipage_after_skip
88 \skip_new:N \g__enumext_minipage_right_skip
89 \skip_new:N \g__enumext_minipage_after_skip
90 \cs_set_protected:Npn \__enumext_tmp:n #1
91   {
92     \dim_new:c  { l__enumext_minipage_left_#1_dim    }
93     \bool_new:c { l__enumext_minipage_active_#1_bool }
94   }
95 \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 (§11.13), the token list `\l__enumext_fake_item_indent_X_tl` is used by itemindent key (§11.16.1), the variables `\l__enumext_label_fill_left_X_tl` and `\l__enumext_label_fill_left_X_tl` are used by the align key (§11.11). 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 (§11.18).

```
96 \cs_set_protected:Npn \__enumext_tmp:n #1
97   {
98     \bool_new:c { l__enumext_wrap_label_#1_bool     }
99     \bool_new:c { l__enumext_wrap_label_opt_#1_bool }
100    \int_new:c  { l__enumext_start_#1_int            }
101    \tl_new:c   { l__enumext_fake_item_indent_#1_tl }
102    \tl_new:c   { l__enumext_label_fill_left_#1_tl  }
103    \tl_new:c   { l__enumext_label_fill_right_#1_tl }
104    \bool_new:c { l__enumext_vspace_a_star_#1_bool  }
105    \bool_new:c { l__enumext_vspace_b_star_#1_bool  }
106  }
107 \clist_map_inline:nn { i, ii, iii, iv, v, vii, viii } { \__enumext_tmp:n {#1} }
```

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

The variable \l__enumext_store_active_bool setting by save-ans key (§11.24.1) activates all the mechanism related to \anskey, anskey*, keyans, keyans* and keyanspic environments.

The variable \l__enumext_store_name_tl saves the {⟨*store name*⟩} set by the save-ans key of the *sequence* and *prop list* in which we will store, the variable \g__enumext_store_name_tl it's just a global copy of {⟨*store name*⟩} used by different functions.

The variable \l__enumext_store_anskey_arg_tl save the *argument* of \anskey (§11.27) and the variables \l__enumext_store_anskey_env_tl and \l__enumext_store_anskey_opt_tl save the ⟨*body*⟩ and the ⟨*keys*⟩ of the environment anskey* (§11.28).

The variables \l__enumext_store_current_label_tl and \l__enumext_store_current_opt_-arg_tl save the *current label* and *optional argument* of \item* (§11.32.2) and \anspic* (§11.37.1) for the keyans, keyans* and keyanspic environments.

The variable \l__enumext_store_current_label_tmp_tl is a temporary variable used by keyans, keyans* and keyanspic at various points.

```
108 \bool_new:N \l__enumext_store_active_bool
109 \tl_new:N   \l__enumext_store_name_tl
110 \tl_new:N   \g__enumext_store_name_tl
111 \tl_new:N   \l__enumext_store_anskey_arg_tl
112 \tl_new:N   \l__enumext_store_anskey_env_tl
113 \tl_new:N   \l__enumext_store_anskey_opt_tl
114 \tl_new:N   \l__enumext_store_current_label_tl
115 \tl_new:N   \l__enumext_store_current_opt_arg_tl
116 \tl_new:N   \l__enumext_store_current_label_tmp_tl
```

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

Internal variables used by the command \setenumext (§11.43).

```
117 \tl_new:N  \l__enumext_setkey_tmpa_tl
118 \tl_new:N  \l__enumext_setkey_tmpb_tl
119 \int_new:N \l__enumext_setkey_tmpa_int
120 \seq_new:N \l__enumext_setkey_tmpa_seq
121 \seq_new:N \l__enumext_setkey_tmpb_seq
```

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

Internal variables used by command \printkeyans (§11.42), show-pos key (§11.26), item-sym* key (§11.30), save-key key (§11.26.2) and *"storage level system"*.

```
122 \tl_new:N  \l__enumext_print_keyans_starred_tl
123 \str_new:N \l__enumext_mark_position_str
124 \tl_new:N  \g__enumext_item_symbol_tl
125 \cs_set_protected:Npn \__enumext_tmp:n #1
126   {
127     \tl_new:c   { l__enumext_print_keyans_#1_tl       }
128     \tl_new:c   { l__enumext_store_save_key_#1_tl      }
129     \bool_new:c { l__enumext_store_save_key_#1_bool    }
130     \bool_new:c { l__enumext_store_upper_level_#1_bool }
131   }
132 \clist_map_inline:nn { i, ii, iii, iv, vii } { \__enumext_tmp:n {#1} }
```

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

Internal variables used by keyanspic environment (§11.37.2).

```
133 \seq_new:N  \l__enumext_keyans_pic_body_seq
134 \dim_new:N  \l__enumext_keyans_pic_width_dim
135 \int_new:N  \l__enumext_keyans_pic_above_int
136 \int_new:N  \l__enumext_keyans_pic_below_int
137 \skip_new:N \l__enumext_keyans_pic_above_skip
```

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

Internal variables used by *"internal check answer"* mechanism (§11.24.3) used by the check-ans and no-store keys and check for starred commands \item* in keyans and keyans* environments and \anspic* in keyanspic environment.

```
138 \bool_new:N \l__enumext_check_answers_bool
139 \bool_new:N \g__enumext_check_ans_key_bool
140 \tl_new:N   \l__enumext_check_start_line_env_tl
141 \int_new:N  \g__enumext_check_starred_cmd_int
142 \int_new:N  \g__enumext_item_anskey_int
```

```
143  \int_new:N   \g__enumext_item_number_int
144  \bool_new:N  \l__enumext_item_number_bool
145  \int_new:N   \g__enumext_item_answer_diff_int
```

(*End of definition for* \l__enumext_check_answers_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 (§11.8). The boolean variable \l__enumext_footnotes_key_bool determine if hyperref is load with key hyperfootnotes=true.

```
146  \bool_new:N \l__enumext_hyperref_bool
147  \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_write_aux_file_tl
\l__enumext_label_copy_X_tl

Internal variables used by save-ref key (§11.26). 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 (§11.8) and the variable \l__enumext_write_aux_file_tl will be in charge of executing the writing code in the .aux file.

```
148  \tl_new:N \l__enumext_newlabel_arg_one_tl
149  \tl_new:N \l__enumext_newlabel_arg_two_tl
150  \tl_new:N \l__enumext_write_aux_file_tl
151  \cs_set_protected:Npn \__enumext_tmp:n #1
152    {
153      \tl_new:c { l__enumext_label_copy_#1_tl }
154    }
155  \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 (§11.31).

```
156  \int_new:N \g__enumext_footnote_int
157  \seq_new:N \g__enumext_footnote_arg_seq
158  \seq_new:N \g__enumext_footnote_int_seq
```

(*End of definition for* \g__enumext_footnote_int, \g__enumext_footnote_arg_seq, *and* \g__enumext_footnote_int_seq.)

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

Internal variables used by enumext* and keyans* environments.

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

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

\c__enumext_all_envs_clist

An internal clist-var variable to run with __enumext_tmp:n.

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

(*End of definition for* \c__enumext_all_envs_clist.)

11.5 Some utility functions

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

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

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

__enumext_after_env:nn
__enumext_before_env:nn — A internal *"hook"* functions for execute code `mini-right` and `mini-right*` keys outside the `enumext*` and `keyans*` environments and print `check-ans` outside the `enumext` and `enumext*` environments.

```
188 \cs_new_protected:Npn \__enumext_after_env:nn #1 #2
189   {
190     \hook_gput_code:nnn {env/#1/after} {enumext} {#2}
191   }
192 \cs_new_protected:Npn \__enumext_before_env:nn #1 #2
193   {
194     \hook_gput_code:nnn {env/#1/before} {enumext} {#2}
195   }
```

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

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

```
196 \cs_new:Nn \__enumext_level:
197   {
198     \int_to_roman:n { \l__enumext_level_int }
199   }
```

(*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?.

```
200 \prg_new_protected_conditional:Npnn \__enumext_if_is_int:n #1 { T, F, TF }
201   {
202     \regex_match:nnTF { ^[\+\-]?[\d]+$ } {#1} % $
203       { \prg_return_true: }
204       { \prg_return_false: }
205   }
```

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

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

```
206 \cs_new_protected:Nn \__enumext_regex_counter_style:
207   {
208     \tl_map_inline:Nn \c__enumext_counter_style_tl
209       {
210         \regex_replace_once:nnN { \c{##1}\* }
211           { \c{##1}\cB{\u{l__enumext_ref_the_count_tl}\cE} } \l__enumext_ref_key_arg_tl
212       }
213   }
```

(*End of definition for* __enumext_regex_counter_style:.)

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

```
214 \cs_new:Npn \__enumext_show_length:nnn #1 #2 #3
215   {
216     * ~ #2
217     \prg_replicate:nn { 14 - \str_count:n {#2} } { ~ }
218       = ~ \use:c { #1_use:c } { l__enumext_#2_#3_#1 } \\
219   }
```

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

11.5.1 Utilities for environments and levels

__enumext_is_not_nested:
__enumext_is_on_first_level:

The function __enumext_is_not_nested: set the variables \g__enumext_standar_bool and \g__enumext_starred_bool to *"true"* only if the environments enumext and enumext* are nested in each other.

```
220  \cs_new_protected:Nn \__enumext_is_not_nested:
221    {
222      \str_case:en { \@currenvir }
223        {
224          {enumext}
225            {
226              \bool_lazy_and:nnT
227                { \bool_not_p:n { \g__enumext_standar_bool } }
228                { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
229                {
230                  \bool_gset_true:N \g__enumext_standar_bool
231                }
232            }
233          {enumext*}
234            {
235              \bool_lazy_and:nnT
236                { \bool_not_p:n { \g__enumext_starred_bool } }
237                { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
238                {
239                  \bool_gset_true:N \g__enumext_starred_bool
240                }
241            }
242        }
243    }
```

The function __enumext_is_on_first_level: will set the variables \l__enumext_standar_first_bool (§11.24.1), \l__enumext_starred_first_bool (§11.24.1) and \l__enumext_anskey_env_bool (§11.28) to *"true"* only if the environment is not nested and we are in the *"first level"* of it . We will also save the *start line number* of each environment in the variable \g__enumext_start_line_tl and the *name* of each environment in the variable \g__enumext_envir_name_tl to use in messages related to the check-ans key and .log file.

```
244  \cs_new_protected:Nn \__enumext_is_on_first_level:
245    {
246      \bool_lazy_all:nT
247        {
248          { \bool_if_p:N \g__enumext_standar_bool }
249          { \int_compare_p:nNn { \l__enumext_level_int } = { 1 } }
250          { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
251        }
252        {
253          \bool_set_true:N \l__enumext_standar_first_bool
254          \bool_set_true:N \l__enumext_anskey_env_bool
255          \tl_gset:Nn \g__enumext_envir_name_tl { enumext }
256          \tl_gset:Ne \g__enumext_start_line_tl
257            {
258              on ~ line ~ \exp_not:V \inputlineno
259            }
260        }
261      \bool_lazy_all:nT
262        {
263          { \bool_if_p:N \g__enumext_starred_bool }
264          { \int_compare_p:nNn { \l__enumext_level_h_int } = { 1 } }
265          { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
266        }
267        {
268          \bool_set_true:N \l__enumext_starred_first_bool
269          \bool_set_true:N \l__enumext_anskey_env_bool
270          \tl_gset:Nn \g__enumext_envir_name_tl { enumext* }
271          \tl_gset:Ne \g__enumext_start_line_tl
272            {
273              on ~ line ~ \exp_not:V \inputlineno
274            }
275        }
276    }
```

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

__enumext_keyans_start_line:

The function `\__enumext_keyans_start_line:` will save the start line number of the environments keyans, keyans* and keyanspic in the variable `\l__enumext _check_start_line_env_tl` to use in the `\__enumext_check_starred_cmd:n` function.

```
277  \cs_new_protected:Nn \__enumext_keyans_start_line:
278    {
279      \str_case:en { \@currenvir }
280        {
281          {keyans}
282            {
283              \tl_set:Ne \l__enumext_check_start_line_env_tl
284                {
285                  in ~ 'keyans' ~ start ~ on ~ line ~ \exp_not:V \inputlineno
286                }
287            }
288          {keyans*}
289            {
290              \tl_set:Ne \l__enumext_check_start_line_env_tl
291                {
292                  in ~ 'keyans*' ~ start ~ on ~ line ~ \exp_not:V \inputlineno
293                }
294            }
295          {keyanspic}
296            {
297              \tl_set:Ne \l__enumext_check_start_line_env_tl
298                {
299                  in ~ 'keyanspic' ~ start ~ on ~ line ~ \exp_not:V \inputlineno
300                }
301            }
302        }
303    }
```

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

11.5.2 Utilities for log and terminal

__enumext_reset_global_vars:
__enumext_reset_global_int:
__enumext_reset_global_bool:
__enumext_reset_global_tl:

The function `\__enumext_reset_global_vars:` will be passed to the function `\__enumext_execute_-after_env:` and will return the global variables to their default values after being used.

```
304  \cs_new_protected:Nn \__enumext_reset_global_vars:
305    {
306      \__enumext_reset_global_int:
307      \__enumext_reset_global_bool:
308      \__enumext_reset_global_tl:
309    }
310  \cs_new_protected:Nn \__enumext_reset_global_int:
311    {
312      \int_gzero:N \g__enumext_item_number_int
313      \int_gzero:N \g__enumext_item_anskey_int
314      \int_gzero:N \g__enumext_item_answer_diff_int
315    }
316  \cs_new_protected:Nn \__enumext_reset_global_bool:
317    {
318      \bool_gset_false:N \g__enumext_check_ans_key_bool
319      \bool_gset_false:N \g__enumext_standar_bool
320      \bool_gset_false:N \g__enumext_starred_bool
321    }
322  \cs_new_protected:Nn \__enumext_reset_global_tl:
323    {
324      \tl_gclear:N \g__enumext_store_name_tl
325      \tl_gclear:N \g__enumext_start_line_tl
326      \tl_gclear:N \g__enumext_envir_name_tl
327    }
```

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

__enumext_log_global_vars:
__enumext_log_answer_vars:

The function `\__enumext_log_global_vars:` will be passed to the function `\__enumext_execute_-after_env:` and write to the .log file the number of elements saved in the ⟨*prop list*⟩ and ⟨*sequence*⟩ created by the save-ans key along with the value of the integer variable created for the resume key.

```
328  \cs_new_protected:Nn \__enumext_log_global_vars:
329    {
330      \msg_log:nneeee { enumext } { prop-seq-int-hook }
331        { \g__enumext_store_name_tl }
```

```
332      { \prop_count:c { g__enumext_ \g__enumext_store_name_tl _prop } }
333      { \seq_count:c { g__enumext_ \g__enumext_store_name_tl _seq } }
334      { \int_use:c { g__enumext_resume_ \g__enumext_store_name_tl _int } }
335    }
```

The function __enumext_log_answer_vars: will be passed to the function __enumext_execute_-
after_env: and write to the .log file the number of items and answers along with the difference between
them.

```
336 \cs_new_protected:Nn \__enumext_log_answer_vars:
337   {
338     \msg_log:nneee { enumext } { item-answer-hook }
339       { \int_use:N \g__enumext_item_number_int }
340       { \int_use:N \g__enumext_item_anskey_int }
341       { \int_eval:n { \g__enumext_item_number_int - \g__enumext_item_anskey_int} }
342   }
```

(*End of definition for* __enumext_log_global_vars: *and* __enumext_log_answer_vars:*.*)

11.6 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.

```
343 \__enumext_at_begin_document:n
344   {
345     \cs_new_eq:NN \__enumext_start_list:nn \list
346     \cs_new_eq:NN \__enumext_stop_list: \endlist
347     \cs_new_eq:NN \__enumext_item_std:w \item
348   }
```

(*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.

```
349 \__enumext_at_begin_document:n
350   {
351     \cs_new_eq:NN \__enumext_minipage:w \minipage
352     \cs_new_eq:NN \__enumext_endminipage: \endminipage
353   }
```

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

11.7 The internal `minipage` environment

__enumext_internal_mini_page:
__enumext_mini_env*

The function __enumext_internal_mini_page: creates a internal __enumext_mini_env* environ-
ment (*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 envi-
ronment will be used internally by the mini-env key, it is not documented in the user interface and is for
internal use only. This function is passed to the function __enumext_safe_exec: in the enumext envi-
ronment definition (§11.35) and __enumext_safe_exec_vii: in the enumext* environment definition
(§11.39)

```
354 \cs_new_protected:Nn \__enumext_internal_mini_page:
355   {
356     \int_compare:nNnT { \l__enumext_level_int } = { 0 }
357       {
358         \DeclareDocumentEnvironment{__enumext_mini_env*}{ m }
359           {
```

```
360            \__enumext_minipage:w [ t ] { ##1 }
361                \legacy_if_gset_false:n { @minipage }
362                \vspace { 0pt }
363            }
364          { \__enumext_endminipage: }
365      }
366    }
```

(*End of definition for* __enumext_internal_mini_page: *and* __enumext_mini_env*.)

11.8 Compatibility with hyperref and footnotehyper

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

```
367 \hook_gput_code:nnn { begindocument } { enumext } { \__enumext_after_hyperref: }
368 \hook_gset_rule:nnnn { begindocument } { enumext } { after } { hyperref }
```

__enumext_after_hyperref:
__enumext_hypertarget:nn
__enumext_phantomsection:

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

```
369 \cs_new_protected:Nn \__enumext_after_hyperref:
370   {
371     \IfPackageLoadedTF { hyperref }
372       {
373         \msg_info:nnn { enumext } { package-load } { hyperref }
374         \bool_set_true:N \l__enumext_hyperref_bool
375         \IfHyperBoolean{hyperfootnotes}
376           {
377             \typeout{hyperfootnotes=true}
378             \bool_set_true:N \l__enumext_footnotes_key_bool
379           }
380           { \typeout{hyperfootnotes=false} }
381       }
382       {  }
```

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.

```
383     \bool_if:NT \l__enumext_footnotes_key_bool
384       {
385         \IfPackageLoadedTF { footnotehyper }
386           {
387             \msg_info:nnn { enumext } { package-load } { footnotehyper }
388           }
389           {
390             \typeout{No ~ footnotehyper ~ load}
391             \typeout{Load ~ and  ~ use  ~ \string\makesavenoteenv{enumext*}}
392             \bool_set_false:N \l__enumext_footnotes_key_bool
393           }
394       }
```

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.

```
395     \bool_if:NTF \l__enumext_hyperref_bool
396       {
397         \cs_new_eq:NN \__enumext_hypertarget:nn \hypertarget
398         \cs_new_eq:NN \__enumext_phantomsection: \phantomsection
399       }
400       {
401         \cs_new_eq:NN \__enumext_hypertarget:nn \use_none:nn
402         \cs_new_eq:NN \__enumext_phantomsection: \prg_do_nothing:
403       }
404   }
```

(*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.

```
405 \cs_new_protected:Npn \__enumext_newlabel:nn #1 #2
406   {
407     \protected@write \@auxout { }
408       {
409         \token_to_str:N \newlabel {#1}
410           {
411             {#2}
412             \bool_if:NT \l__enumext_hyperref_bool
413               { { \thepage } {#2} {#1} }
414             { }
415           }
416       }
417     \__enumext_hypertarget:nn {#1} { }
418     \__enumext_phantomsection:
419   }
```

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

11.9 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.

```
420 \cs_new_protected:Npn \__enumext_define_counters:Nn #1 #2
421   {
422     \cs_if_exist:cTF { c@ #2 }
423       { \msg_fatal:nnn { enumext } { counters }{ #2 } }
424       {
425         \tl_set:Nn #1 { #2 }
426         \newcounter { #2 }
427       }
428   }
```

(*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.

```
429 \__enumext_define_counters:Nn \l__enumext_counter_i_tl    { enumXi     }
430 \__enumext_define_counters:Nn \l__enumext_counter_ii_tl   { enumXii    }
431 \__enumext_define_counters:Nn \l__enumext_counter_iii_tl  { enumXiii   }
432 \__enumext_define_counters:Nn \l__enumext_counter_iv_tl   { enumXiv    }
433 \__enumext_define_counters:Nn \l__enumext_counter_v_tl    { enumXv     }
434 \__enumext_define_counters:Nn \l__enumext_counter_vi_tl   { enumXvi    }
435 \__enumext_define_counters:Nn \l__enumext_counter_vii_tl  { enumXvii   }
436 \__enumext_define_counters:Nn \l__enumext_counter_viii_tl { enumXviii  }
```

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

11.10 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.

```
437 \cs_new_protected:Npn \__enumext_register_counter_style:Nn #1 #2
438   {
439     \tl_const:cn { c__enumext_widest_ \cs_to_str:N #1 _tl } {#2}
440     \tl_gput_right:Nn \g__enumext_counter_styles_tl {#1}
441   }
442 \__enumext_register_counter_style:Nn \arabic { 0 }
443 \__enumext_register_counter_style:Nn \Alph   { M }
444 \__enumext_register_counter_style:Nn \alph   { m }
445 \__enumext_register_counter_style:Nn \Roman  { VIII }
446 \__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.

```
447 \cs_new_protected:Npn \__enumext_label_width_by_box:Nn #1 #2
448   {
449     \hbox_set:Nn \l__enumext_label_width_by_box {#2}
450     \dim_set:Nn #1 { \box_wd:N \l__enumext_label_width_by_box }
451   }
452 \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.

```
453 \cs_new_protected:Npn \__enumext_label_style:Nnn #1 #2 #3
454   {
455     \tl_clear_new:N #1
456     \tl_put_right:Ne #1 { \tl_trim_spaces:n {#3} }
457     \tl_gset_eq:NN \g__enumext_widest_label_tl #1
458     \tl_map_inline:Nn \g__enumext_counter_styles_tl
459       {
460         \tl_replace_all:Nne #1 { ##1* } { \exp_not:N ##1 {#2} }
461         \tl_greplace_all:Nne \g__enumext_widest_label_tl { ##1* }
462           { \tl_use:c { c__enumext_widest_ \cs_to_str:N ##1 _tl } }
463       }
464     \__enumext_label_width_by_box:Nn \l__enumext_current_widest_dim
465       { \tl_use:N \g__enumext_widest_label_tl }
466     \tl_set_eq:cN { the #2 } #1
467   }
468 \cs_generate_variant:Nn \__enumext_label_style:Nnn { cvn }
```

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

11.11 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.

```
469 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
470   {
471     \keys_define:nn { enumext / #1 }
472       {
473         font       .tl_set:c   = { l__enumext_label_font_style_#2_tl },
474         font       .value_required:n = true,
475         labelsep   .dim_set:c  = { l__enumext_labelsep_#2_dim },
476         labelsep   .initial:n  = {0.3333em},
477         labelsep   .value_required:n = true,
478         labelwidth .dim_set:c  = { l__enumext_labelwidth_#2_dim },
479         labelwidth .value_required:n = true,
480         wrap-label .cs_set_protected:cp = { __enumext_wrapper_label_#2:n } ##1,
481         wrap-label .initial:n  = {##1},
482         wrap-label .value_required:n = true,
483         wrap-label* .code:n = {
484                       \bool_set_true:c { l__enumext_wrap_label_opt_#2_bool }
485                       \keys_set:nn { enumext / #1 } { wrap-label = {##1} }
486                     },
487         wrap-label* .value_required:n = true,
488       }
489   }
490 \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.

```
491  \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
492    {
493      \keys_define:nn { enumext / #1 }
494        {
495          align .choice:,
496          align / left     .code:n =
497                            {
498                              \tl_clear:c { l__enumext_label_fill_left_#2_tl  }
499                              \tl_set:cn  { l__enumext_label_fill_right_#2_tl } { \hfill }
500                            },
501          align / right    .code:n =
502                            {
503                              \tl_set:cn  { l__enumext_label_fill_left_#2_tl  } { \hfill }
504                              \tl_clear:c { l__enumext_label_fill_right_#2_tl }
505                            },
506          align / center   .code:n =
507                            {
508                              \tl_set:cn { l__enumext_label_fill_left_#2_tl  } { \hfill }
509                              \tl_set:cn { l__enumext_label_fill_right_#2_tl } { \hfill }
510                            },
511          align / unknown .code:n =
512                              \msg_error:nneee { enumext } { unknown-choice }
513                                { align } { left, ~ right, ~  center } { \exp_not:n {##1} },
514          align .initial:n  = left,
515          align .value_required:n  = true,
516        }
517    }
518  \clist_map_inline:nn
519    {
520      {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {keyans}{v}
521    }
522    { \__enumext_tmp:nn #1 }

523  \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
524    {
525      \keys_define:nn { enumext / #1 }
526        {
527          align .choice:,
528          align / left     .code:n = \str_set:cn { l__enumext_align_label_#2_str } { l },
529          align / right    .code:n = \str_set:cn { l__enumext_align_label_#2_str } { r },
530          align / center   .code:n = \str_set:cn { l__enumext_align_label_#2_str } { c },
531          align / unknown .code:n =
532                              \msg_error:nneee { enumext } { unknown-choice }
533                                { align } { left, ~ right, ~  center } { \exp_not:n {##1} },
534          align .initial:n = left,
535          align .value_required:n = true,
536        }
537    }
538  \clist_map_inline:nn { {enumext*}{vii}, {keyans*}{viii} } { \__enumext_tmp:nn #1 }
```

(*End of definition for* align.)

11.12 Setting `label` and `ref` keys

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

11.12.1 Define and set `label` and `ref` keys for enumext environment

label
ref
\l__enumext_label_i_tl
\l__enumext_label_ii_tl
\l__enumext_label_iii_tl
\l__enumext_label_iv_tl

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

```
539  \cs_set_protected:Npn \__enumext_tmp:nnn #1 #2 #3
540    {
541      \keys_define:nn { enumext / #1 }
542        {
543          label .code:n    = {
544                                \__enumext_label_style:cvn { l__enumext_label_#2_tl }
545                                  { l__enumext_counter_#2_tl } {##1}
546                                \dim_set_eq:cN  { l__enumext_labelwidth_#2_dim }
547                                  \l__enumext_current_widest_dim
```

```
548                          },
549            label  .initial:n = #3,
550            label  .value_required:n = true,
551            ref    .code:n      = \__enumext_standar_ref:n {##1},
552            ref    .value_required:n = true,
553          }
554    }
555  \__enumext_tmp:nnn { level-1 } {   i } { \arabic*.}
556  \__enumext_tmp:nnn { level-2 } {  ii } { (\alph*) }
557  \__enumext_tmp:nnn { level-3 } { iii } { \roman*. }
558  \__enumext_tmp:nnn { level-4 } {  iv } { \Alph*.  }
```

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

__enumext_standar_ref:n
__enumext_standar_ref:

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

```
559  \cs_new_protected:Npn \__enumext_standar_ref:n #1
560    {
561      \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
562      \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
563        {
564          \msg_error:nnn { enumext } { key-ref-empty } { enumext }
565        }
566        {
567          \tl_set_eq:Nc
568            \l__enumext_ref_the_count_tl { l__enumext_counter_ \__enumext_level: _tl }
569          \__enumext_regex_counter_style:
570          \tl_set_eq:Nc
571            \l__enumext_ref_the_count_tl { l__enumext_the_counter_ \__enumext_level: _tl }
572          \tl_put_right:ce { l__enumext_renew_the_count_ \__enumext_level: _tl }
573            {
574              \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl }
575                { \exp_not:V \l__enumext_ref_key_arg_tl }
576            }
577        }
578    }
```

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

```
579  \cs_new_protected:Nn \__enumext_standar_ref:
580    {
581      \tl_if_empty:cF { l__enumext_renew_the_count_ \__enumext_level: _tl }
582        {
583          \tl_use:c { l__enumext_renew_the_count_ \__enumext_level: _tl }
584        }
585    }
```

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

11.12.2 Define and set label and ref keys for enumext* and keyans* environments

label
ref

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

\l__enumext_label_vii_tl
\l__enumext_label_viii_tl

```
586  \cs_set_protected:Npn \__enumext_tmp:nnn #1 #2 #3
587    {
588      \keys_define:nn { enumext / #1 }
589        {
590          label .code:n    = {
591                               \__enumext_label_style:cvn { l__enumext_label_#2_tl }
592                                 { l__enumext_counter_#2_tl } {##1}
593                               \dim_set_eq:cN  { l__enumext_labelwidth_#2_dim }
594                                 \l__enumext_current_widest_dim
595                             },
596          label .initial:n = #3,
597          label .value_required:n = true,
598          ref   .code:n    = \__enumext_starred_ref:n {##1},
599          ref   .value_required:n = true,
600        }
```

```
601      }
602  \__enumext_tmp:nnn { enumext* } {  vii } { \arabic*.}
603  \__enumext_tmp:nnn { keyans*  } { viii } { \Alph*) }
```

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

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

```
604  \cs_new_protected:Npn \__enumext_starred_ref:n #1
605    {
606      \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
607      \int_compare:nNnT { \l__enumext_level_h_int } = { 1 }
608        {
609          \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
610            {
611              \msg_error:nnn { enumext } { key-ref-empty } { enumext* }
612            }
613            {
614              \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_counter_vii_tl
615              \__enumext_regex_counter_style:
616              \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_the_counter_vii_tl
617              \tl_put_right:Ne \l__enumext_renew_the_count_vii_tl
618                {
619                  \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl }
620                    { \exp_not:V \l__enumext_ref_key_arg_tl }
621                }
622            }
623        }
624      \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
625        {
626          \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
627            {
628              \msg_error:nnn { enumext } { key-ref-empty } { keyans* }
629            }
630            {
631              \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_counter_viii_tl
632              \__enumext_regex_counter_style:
633              \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_the_counter_viii_tl
634              \tl_put_right:Ne \l__enumext_renew_the_count_viii_tl
635                {
636                  \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl }
637                    { \exp_not:V \l__enumext_ref_key_arg_tl }
638                }
639            }
640        }
641    }
```

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

```
642  \cs_new_protected:Nn \__enumext_starred_ref:
643    {
644      \int_compare:nNnT { \l__enumext_level_h_int } = { 1 }
645        {
646          \tl_if_empty:NF \l__enumext_renew_the_count_vii_tl
647            {
648              \tl_use:N \l__enumext_renew_the_count_vii_tl
649            }
650        }
651      \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
652        {
653          \tl_if_empty:NF \l__enumext_renew_the_count_viii_tl
654            {
655              \tl_use:N \l__enumext_renew_the_count_viii_tl
656            }
657        }
658    }
```

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

11.12.3 Define and set `label` and `ref` keys for `keyans` and `keyanspic` environments

label
ref
\l__enumext_label_v_tl
\l__enumext_label_vi_tl

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

```
659 \keys_define:nn { enumext / keyans }
660   {
661     label .code:n    = {
662                          \__enumext_label_style:cvn { l__enumext_label_v_tl }
663                            { l__enumext_counter_v_tl } {#1}
664                          \dim_set_eq:cN  { l__enumext_labelwidth_v_dim }
665                            \l__enumext_current_widest_dim
666                          \__enumext_label_style:cvn { l__enumext_label_vi_tl }
667                            { l__enumext_counter_vi_tl } {#1}
668                          \dim_set_eq:cN  { l__enumext_labelwidth_v_dim }
669                            \l__enumext_current_widest_dim
670                        },
671     label .initial:n = \Alph*),
672     label .value_required:n = true,
673     ref   .code:n    = \__enumext_keyans_ref:n {#1},
674     ref   .value_required:n = true,
675   }
```

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

__enumext_keyans_ref:n
__enumext_keyans_ref:

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

```
676 \cs_new_protected:Npn \__enumext_keyans_ref:n #1
677   {
678     \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
679     \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
680       {
681         \msg_error:nnn { enumext } { key-ref-empty } { keyans }
682       }
683       {
684         \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_counter_v_tl
685         \__enumext_regex_counter_style:
686         \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_the_counter_v_tl
687         \tl_put_right:Ne \l__enumext_renew_the_count_v_tl
688           {
689             \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl }
690               { \exp_not:V \l__enumext_ref_key_arg_tl }
691           }
692       }
693   }
```

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

```
694 \cs_new_protected:Nn \__enumext_keyans_ref:
695   {
696     \tl_if_empty:NF \l__enumext_renew_the_count_v_tl
697       {
698         \tl_use:N \l__enumext_renew_the_count_v_tl
699       }
700   }
```

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

11.13 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.

```
701 \cs_new_protected:Npn \__enumext_start_from:NNn #1 #2 #3
702   {
703     \__enumext_if_is_int:nTF { #3 }
704       {
705         \int_set:Nn #2 {#3}
706       }
```

```
707        {
708          \regex_match:nVT { \c{Alph} | \c{alph} } {#1}
709            { \int_set:Nn #2 { \int_from_alph:n {#3} } }
710          \regex_match:nVT { \c{Roman} | \c{roman} } {#1}
711            { \int_set:Nn #2  { \int_from_roman:n {#3} } }
712        }
713     }
714   \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.

```
715   \cs_new_protected:Npn \__enumext_widest_from:nNNn #1 #2 #3 #4
716     {
717        \__enumext_if_is_int:nTF {#4}
718          {
719            \setcounter{enumX#1} { #4 }
720          }
721          {
722            \regex_match:nVT { \c{Alph} | \c{alph} } {#2}
723              { \setcounter{enumX#1} { \int_from_alph:n {#4} } }
724            \regex_match:nVT { \c{Roman} | \c{roman} } {#2}
725              { \setcounter{enumX#1} { \int_from_roman:n {#4} } }
726          }
727        \__enumext_label_width_by_box:cv
728          { l__enumext_labelwidth_#1_dim } { l__enumext_label_#1_tl }
729     }
730   \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, enumext*, keyans and keyans* environments.

```
731   \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
732     {
733        \keys_define:nn { enumext / #1 }
734          {
735            start  .code:n    = {
736                                  \__enumext_start_from:ccn
737                                    { l__enumext_label_#2_tl }
738                                    { l__enumext_start_#2_int } {##1}
739                                 },
740            start  .initial:n = 1,
741            widest .code:n    = {
742                                  \__enumext_widest_from:nccn {#2}
743                                    { l__enumext_label_#2_tl }
744                                    { l__enumext_labelwidth_#2_dim } {##1}
745                                 },
746            widest .value_required:n = true,
747            start  .value_required:n = true,
748          }
749     }
750   \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*.*)

11.14   Setting keys for vertical spaces

topsep
partopsep
parsep
noitemsep
nosep

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

```
751   \cs_set_protected:Npn \__enumext_tmp:nnnnnn #1 #2 #3 #4 #5 #6
752     {
753        \keys_define:nn { enumext / #1 }
```

```
754      {
755          topsep     .skip_set:c = { l__enumext_topsep_#2_skip },
756          topsep     .initial:n  = {#3},
757          topsep     .value_required:n = true,
758          partopsep .skip_set:c = { l__enumext_partopsep_#2_skip },
759          partopsep .initial:n  = {#4},
760          partopsep .value_required:n = true,
761          parsep     .skip_set:c = { l__enumext_parsep_#2_skip },
762          parsep     .initial:n  = {#5},
763          parsep     .value_required:n = true,
764          itemsep    .skip_set:c = { l__enumext_itemsep_#2_skip },
765          itemsep    .initial:n  = {#6},
766          itemsep    .value_required:n = true,
767          noitemsep .meta:n      = { itemsep = 0pt, parsep = 0pt },
768          noitemsep .value_forbidden:n = true,
769          nosep      .meta:n      = {
770                                     itemsep = 0pt, parsep= 0pt,
771                                     topsep = 0pt, partopsep = 0pt,
772                                   },
773          nosep      .value_forbidden:n = true,
774      }
775   }
```

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

```
776 \__enumext_tmp:nnnnnn { level-1 } { i } { 8.0pt plus 2.0pt minus 4.0pt }
777   { 2.0pt plus 1.0pt minus 1.0pt } { 4.0pt plus 2.0pt minus 1.0pt }
778   { 4.0pt plus 2.0pt minus 1.0pt }
779 \__enumext_tmp:nnnnnn { level-2 } { ii } { 4.0pt plus 2.0pt minus 1.0pt }
780   { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
781   { 2.0pt plus 1.0pt minus 1.0pt }
782 \__enumext_tmp:nnnnnn { level-3 } { iii } { 2.0pt plus 1.0pt minus 1.0pt }
783   { 1.0pt minus 1.0pt }{ 0pt }{ 2.0pt plus 1.0pt minus 1.0pt }
784 \__enumext_tmp:nnnnnn { level-4 } { iv } { 2.0pt plus 1.0pt minus 1.0pt }
785   { 1.0pt minus 1.0pt }{ 0pt }{ 2.0pt plus 1.0pt minus 1.0pt }
786 \__enumext_tmp:nnnnnn { keyans  } { v }{ 4.0pt plus 2.0pt minus 1.0pt }
787   { 2.0pt plus 1.0pt minus 1.0pt }{ 2.0pt plus 1.0pt minus 1.0pt }
788   { 2.0pt plus 1.0pt minus 1.0pt }
789 \__enumext_tmp:nnnnnn { enumext* } { vii } { 8.0pt plus 2.0pt minus 4.0pt }
790   { 2.0pt plus 1.0pt minus 1.0pt } { 4.0pt plus 2.0pt minus 1.0pt }
791   { 4.0pt plus 2.0pt minus 1.0pt }
792 \__enumext_tmp:nnnnnn { keyans* } { viii } { 4.0pt plus 2.0pt minus 1.0pt }
793   { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
794   { 2.0pt plus 1.0pt minus 1.0pt }
```

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

11.15   Setting base-fix key

When nesting starting right after `\item` (without material between them) there is a problem with the alignment of the baseline between the two environments. One way to get around this problem is to place `\mode_leave_vertical:` and then apply `\vspace{-\baselineskip}` and set `topsep=0pt` for the *"first level"* of the nested `enumext` or `enumext*` environments.

base-fix
__enumext_nested_base_line_fix:

We define the key `base-fix` only for the *"first level"* of `enumext` and `enumext*`.

```
795 \cs_set_protected:Npn \__enumext_tmp:n #1
796   {
797     \keys_define:nn { enumext / #1 }
798       {
799         base-fix .bool_set:N = \l__enumext_base_line_fix_bool,
800         base-fix .initial:n  = false,
801         base-fix .value_forbidden:n = true,
802       }
803   }
804 \clist_map_inline:nn { level-1, enumext* } { \__enumext_tmp:n {#1} }
```

The function `\__enumext_nested_base_line_fix:` will be in charge of applying the baseline correction and adjusting the ⟨*keys*⟩. This function is passed to the function `\__enumext_parse_keys:n` in the `enumext` environment definition (§11.35) and to the function `\__enumext_parse_keys_vii:n` in the `enumext*` environment definition (§11.39)

```
805 \cs_new_protected:Nn \__enumext_nested_base_line_fix:
806   {
807     \bool_lazy_and:nnT
```

```
808        { \bool_if_p:N \l__enumext_standar_first_bool }
809        { \bool_if_p:N \l__enumext_base_line_fix_bool }
810        {
811          \mode_leave_vertical:
812          \vspace { -\baselineskip }
813          \keys_set:nn { enumext / level-1 }
814            {
815              topsep = 0pt, above = 0pt, above* = 0pt,
816            }
817        }
818      \bool_lazy_and:nnT
819        { \bool_if_p:N \l__enumext_starred_first_bool }
820        { \bool_if_p:N \l__enumext_base_line_fix_bool }
821        {
822          \mode_leave_vertical:
823          \vspace { -\baselineskip }
824          \keys_set:nn { enumext / enumext* }
825            {
826              topsep = 0pt, above = 0pt, above* = 0pt,
827            }
828        }
829      \bool_set_false:N \l__enumext_base_line_fix_bool
830    }
```

💣 This key is enabled by default in the command \printkeyans (§11.42).

(*End of definition for* base-fix *and* __enumext_nested_base_line_fix:.)

11.16 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, enumext*, keyans and keyans* environments.

```
831 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
832   {
833     \keys_define:nn { enumext / #1 }
834       {
835         itemindent   .dim_set:c = { l__enumext_fake_item_indent_#2_dim },
836         itemindent   .value_required:n = true,
837         rightmargin  .dim_set:c = { l__enumext_rightmargin_#2_dim },
838         rightmargin  .value_required:n = true,
839         listparindent .dim_set:c = { l__enumext_listparindent_#2_dim },
840         listparindent .value_required:n = true,
841         list-offset  .dim_set:c = { l__enumext_listoffset_#2_dim },
842         list-offset  .value_required:n = true,
843         list-indent  .code:n    =
844                       \bool_set_true:c { l__enumext_leftmargin_tmp_#2_bool }
845                       \dim_set:cn { l__enumext_leftmargin_tmp_#2_dim } {##1},
846         list-indent  .value_required:n = true,
847       }
848   }
849 \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.

```
850 \cs_set_protected:Npn \__enumext_tmp:n #1
851   {
852     \keys_define:nn { enumext / #1 } { list-indent .initial:n = 0pt, }
853   }
854 \clist_map_inline:nn { enumext*, keyans* } { \__enumext_tmp:n {#1} }
```

11.16.1 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.

```
855 \cs_set_protected:Nn \__enumext_fake_item:
856   {
857     \dim_compare:nNnT
858       { \dim_use:c { l__enumext_fake_item_indent_ \__enumext_level: _dim } }
859       >
860       { \c_zero_dim }
```

```
861       {
862         \tl_set:ce { l__enumext_fake_item_indent_ \__enumext_level: _tl }
863           {
864             \exp_not:N \mode_leave_vertical:
865             \exp_not:n { \skip_horizontal:n }
866               { \dim_use:c { l__enumext_fake_item_indent_ \__enumext_level: _dim } }
867             \ignorespaces
868           }
869       }
870     }
871 \cs_set_protected:Nn \__enumext_keyans_fake_item:
872   {
873     \dim_compare:nNnT
874       { \l__enumext_fake_item_indent_v_dim } > { \c_zero_dim }
875       {
876         \tl_set:Ne \l__enumext_fake_item_indent_v_tl
877           {
878             \exp_not:N \mode_leave_vertical:
879             \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_v_dim
880           }
881       }
882   }
883 \cs_set_protected:Nn \__enumext_fake_item_vii:
884   {
885     \dim_compare:nNnT
886       { \l__enumext_fake_item_indent_vii_dim } > { \c_zero_dim }
887       {
888         \tl_set:Ne \l__enumext_fake_item_indent_vii_tl
889           {
890             \exp_not:N \mode_leave_vertical:
891             \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_vii_dim
892           }
893       }
894   }
895 \cs_set_protected:Nn \__enumext_fake_item_viii:
896   {
897     \dim_compare:nNnT
898       { \l__enumext_fake_item_indent_viii_dim } > { \c_zero_dim }
899       {
900         \tl_set:Ne \l__enumext_fake_item_indent_viii_tl
901           {
902             \exp_not:N \mode_leave_vertical:
903             \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_viii_dim
904           }
905       }
906   }
```

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

11.17 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.

```
907 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
908   {
909     \keys_define:nn { enumext / #1 }
910       {
911         show-length .bool_set:c = { l__enumext_show_length_#2_bool },
912         show-length .initial:n  = false,
913       }
914   }
915 \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }
```

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

11.18 Setting before, after and first keys

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

```
916 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
```

```
917  {
918    \keys_define:nn { enumext / #1 }
919      {
920        before  .tl_set:c   = { l__enumext_before_no_starred_key_#2_tl },
921        before  .value_required:n = true,
922        before* .tl_set:c   = { l__enumext_before_starred_key_#2_tl },
923        before* .value_required:n = true,
924        after   .tl_set:c   = { l__enumext_after_stop_list_#2_tl },
925        after   .value_required:n = true,
926        first   .tl_set:c   = { l__enumext_after_list_args_#2_tl },
927        first   .value_required:n = true,
928      }
929  }
930  \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }
```

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

11.18.1   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.

```
931  \cs_new_protected:Nn \__enumext_before_args_exec:
932    {
933      \tl_use:c { l__enumext_before_starred_key_ \__enumext_level: _tl }
934    }
```

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*⟩.

```
935  \cs_new_protected:Nn \__enumext_before_keys_exec:
936    {
937      \tl_use:c { l__enumext_before_no_starred_key_ \__enumext_level: _tl }
938    }
```

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

```
939  \cs_new_protected:Nn \__enumext_after_stop_list:
940    {
941      \tl_use:c { l__enumext_after_stop_list_ \__enumext_level: _tl }
942    }
```

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.

```
943  \cs_new_protected:Nn \__enumext_after_args_exec:
944    {
945      \tl_use:c { l__enumext_after_list_args_ \__enumext_level: _tl }
946    }
```

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

11.18.2   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.

```
947  \cs_new_protected:Nn \__enumext_before_args_exec_v:
948    {
949      \tl_use:N \l__enumext_before_starred_key_v_tl
950    }
```

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*⟩.

```
951  \cs_new_protected:Nn \__enumext_before_keys_exec_v:
952    {
953      \tl_use:N \l__enumext_before_no_starred_key_v_tl
954    }
```

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

```
955 \cs_new_protected:Nn \__enumext_after_stop_list_v:
956   {
957     \tl_use:N \l__enumext_after_stop_list_v_tl
958   }
```

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.

```
959 \cs_new_protected:Nn \__enumext_after_args_exec_v:
960   {
961     \tl_use:N \l__enumext_after_list_args_v_tl
962   }
```

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

11.18.3 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.

```
963 \cs_new_protected:Nn \__enumext_before_args_exec_vii:
964   {
965     \tl_use:N \l__enumext_before_starred_key_vii_tl
966   }
967 \cs_new_protected:Nn \__enumext_before_args_exec_viii:
968   {
969     \tl_use:N \l__enumext_before_starred_key_viii_tl
970   }
```

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*⟩.

```
971 \cs_new_protected:Nn \__enumext_before_keys_exec_vii:
972   {
973     \tl_use:N \l__enumext_before_no_starred_key_vii_tl
974   }
975 \cs_new_protected:Nn \__enumext_before_keys_exec_viii:
976   {
977     \tl_use:N \l__enumext_before_no_starred_key_viii_tl
978   }
```

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

```
979 \cs_new_protected:Nn \__enumext_after_stop_list_vii:
980   {
981     \tl_use:N \l__enumext_after_stop_list_vii_tl
982   }
983 \cs_new_protected:Nn \__enumext_after_stop_list_viii:
984   {
985     \tl_use:N \l__enumext_after_stop_list_viii_tl
986   }
```

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.

```
987 \cs_new_protected:Nn \__enumext_after_args_exec_vii:
988   {
989     \tl_use:N \l__enumext_after_list_args_vii_tl
990   }
991 \cs_new_protected:Nn \__enumext_after_args_exec_viii:
992   {
993     \tl_use:N \l__enumext_after_list_args_viii_tl
994   }
```

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

11.19 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`, `enumext*`, `keyans` and `keyans*` environments.

```
995  \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
996    {
997      \keys_define:nn { enumext / #1 }
998        {
999          mini-env    .dim_set:c  = { l__enumext_minipage_right_#2_dim },
1000         mini-env    .value_required:n = true,
1001         mini-sep    .dim_set:c  = { l__enumext_minipage_hsep_#2_dim },
1002         mini-sep    .initial:n  = 0.3333em,
1003         mini-sep    .value_required:n = true,
1004         columns-sep .dim_set:c  = { l__enumext_columns_sep_#2_dim },
1005         columns-sep .value_required:n = true,
1006         columns     .int_set:c  = { l__enumext_columns_#2_int },
1007         columns     .initial:n  = 1,
1008         columns     .value_required:n = true,
1009        }
1010     }
1011 \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 command `\miniright` is not available, so we will add the keys `mini-right` and `mini-right*` to implement support for `minipage` environment.

```
1012 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1013    {
1014      \keys_define:nn { enumext / #1 }
1015        {
1016          mini-right  .tl_gset:c = { g__enumext_miniright_code_#2_tl },
1017          mini-right  .value_required:n = true,
1018          mini-right* .code:n    = {
1019                                    \bool_gset_true:c { g__enumext_minipage_center_#2_bool }
1020                                    \keys_set:nn { enumext / #1 } { mini-right = {##1} }
1021                                   },
1022          mini-right* .value_required:n = true,
1023        }
1024     }
1025 \clist_map_inline:nn { {enumext*}{vii}, {keyans*}{viii} } { \__enumext_tmp:nn #1 }
```

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

11.20 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.
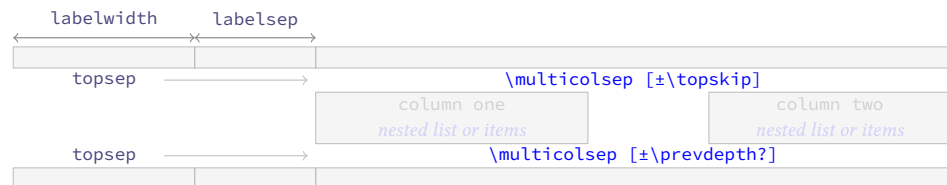
Figure 7: Representation of the vertical space in `multicols` for a nested level.

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.

11.20.1 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*.

```
1026  \cs_new_protected:Nn \__enumext_multi_set_vskip:
1027    {
1028      \skip_set:cn { l__enumext_multicols_above_ \__enumext_level: _skip }
1029        {
1030          \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
1031        }
1032      \skip_set:cn { l__enumext_multicols_below_ \__enumext_level: _skip }
1033        {
1034          \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
1035        }
1036      \__enumext_add_pre_parsep:
1037    }
```

(*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*.

```
1038  \cs_new_protected:Nn \__enumext_add_pre_parsep:
1039    {
1040      \int_case:nn { \l__enumext_level_int }
1041        {
1042          { 2 }{
1043              \skip_if_eq:nnF { \l__enumext_parsep_i_skip } { \c_zero_skip }
1044                {
1045                  \skip_add:Nn \l__enumext_multicols_above_ii_skip { \l__enumext_parsep_i_skip }
1046                }
1047            }
1048          { 3 }{
1049              \skip_if_eq:nnF { \l__enumext_parsep_ii_skip } { \c_zero_skip }
1050                {
1051                  \skip_add:Nn \l__enumext_multicols_above_iii_skip { \l__enumext_parsep_ii_skip }
1052                }
1053            }
1054          { 4 }{
1055              \skip_if_eq:nnF { \l__enumext_parsep_iii_skip } { \c_zero_skip }
1056                {
1057                  \skip_add:Nn \l__enumext_multicols_above_iv_skip { \l__enumext_parsep_iii_skip }
1058                }
1059            }
1060        }
1061    }
```

(*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*⟩.

```
1062  \cs_new_protected:Nn \__enumext_multi_addvspace:
1063    {
1064      \__enumext_multi_set_vskip:
1065      \mode_if_vertical:T
1066        {
1067          \skip_add:cn { l__enumext_multicols_above_ \__enumext_level: _skip }
1068            {
1069              \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
1070            }
1071          \skip_add:cn { l__enumext_multicols_below_ \__enumext_level: _skip }
1072            {
1073              \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
1074            }
1075        }
1076      \par\nopagebreak
1077      \addvspace{ \skip_use:c { l__enumext_multicols_above_ \__enumext_level: _skip } }
1078    }
```

(*End of definition for* __enumext_multi_addvspace:.)

11.20.2   Adjustment of vertical spaces for `multicols` in keyans

`\__enumext_keyans_multi_set_vskip:`
`\__enumext_keyans_multi_addvspace:`

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

```
1079  \cs_new_protected:Nn \__enumext_keyans_multi_set_vskip:
1080    {
1081      \skip_set:Nn \l__enumext_multicols_above_v_skip
1082        {
1083          \l__enumext_topsep_v_skip
1084        }
1085      \skip_set:Nn \l__enumext_multicols_below_v_skip
1086        {
1087          \l__enumext_topsep_v_skip
1088        }
1089    }
1090  \cs_new_protected:Nn \__enumext_keyans_multi_addvspace:
1091    {
1092      \__enumext_keyans_multi_set_vskip:
1093      \mode_if_vertical:T
1094        {
1095          \skip_add:Nn \l__enumext_multicols_above_v_skip
1096            {
1097              \skip_use:N \l__enumext_partopsep_v_skip
1098            }
1099          \skip_add:Nn \l__enumext_multicols_below_v_skip
1100            {
1101              \skip_use:N \l__enumext_partopsep_v_skip
1102            }
1103        }
1104      \par\nopagebreak
1105      \addvspace{ \l__enumext_multicols_above_v_skip }
1106    }
```

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

11.21   Adjustment of vertical spaces for `minipage`

When nesting a *"list environment"* within the `minipage` environment, the values of the *"vertical spaces"* are lost. Graphically it can be seen like in the figure 8.
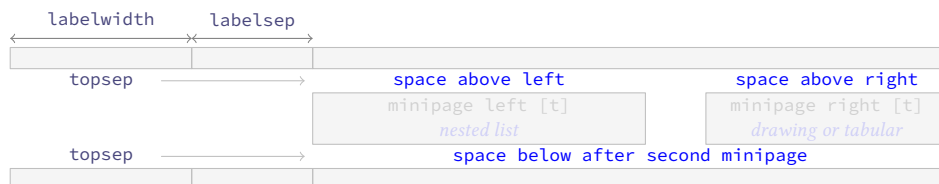
Figure 8: Representation of the `minipage` spacing adjustment for a nested level.

Since we want to keep the *"left"* and *"right"* environments *"aligned on top"*, preserving the `\baselineskip` and keep the desired *"spaces"* (`\topsep` + [`\partopsep`]) it is necessary to *"adjust"* the *"vertical spaces"* for `minipage` environments.

Here there are several complications that we must circumvent, the `minipage` environment eliminates the "top" spaces, the `multicols` environment can be nested in the `minipage` environment, the "top" and "bottom" spaces are affected when `topsep=0pt` and to this is added the `\partopsep` parameter that comes into action according to whether TeX is in ⟨*horizontal mode*⟩ or ⟨*vertical mode*⟩. Depending on these cases, small adjustments must be made using `\vspace` and `\addvspace` to obtain the *"desired vertical spacing"*.

💣 Again I must make clear that the implementation here is a *"bit questionable"*, but hunting the spaces (`glue`) produced by the `minipage` environment is quite complicated, even more if `multicols` it is nested. The setting of the values was more *"trial and error"* (aprox to `\strutbox`), using the help of the `lua-visual-debug`[14] package, again my attempts to find the correct values using `\showoutput` and `\showboxdepth` absolutely failed.

11.21.1   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`.

```
1107  \cs_new_protected:Nn \__enumext_mini_set_vskip:
1108    {
1109      \int_compare:nNnTF
1110        { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } > { 1 }
1111        {
```

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.

```
1112          \skip_if_eq:nnTF
1113            { \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip } } { \c_zero_skip }
1114            {
1115              \skip_set:Nn \l__enumext_minipage_left_skip
1116                {
1117                  -0.150\box_dp:N \strutbox
1118                }
1119              \skip_set:Nn \l__enumext_minipage_right_skip
1120                {
1121                  0.695\box_dp:N \strutbox
1122                }
1123              \skip_set:Nn \l__enumext_minipage_after_skip
1124                {
1125                  \box_dp:N \strutbox
1126                }
1127              \__enumext_zero_parsep:
1128            }
1129            {
1130              \skip_set:Nn \l__enumext_minipage_left_skip
1131                {
1132                  \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
1133                }
1134              \skip_set:Nn \l__enumext_minipage_right_skip
1135                {
1136                  0.695\box_dp:N \strutbox
1137                }
1138              \skip_set:Nn \l__enumext_minipage_after_skip
1139                {
1140                  1.85\box_dp:N \strutbox
1141                  + \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
1142                }
1143            }
1144        }
1145        {
```

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.

```
1146          \skip_if_eq:nnTF
1147            { \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip } } { \c_zero_skip }
1148            {
1149              \skip_set:Nn \l__enumext_minipage_left_skip
1150                {
1151                  0.5\box_dp:N \strutbox
1152                  - \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
1153                }
1154              \skip_set:Nn \l__enumext_minipage_right_skip
1155                {
1156                  \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
1157                }
1158              \skip_set:Nn \l__enumext_minipage_after_skip
1159                {
1160                  1.6\box_dp:N \strutbox
1161                }
1162            }
1163            {
1164              \skip_set:Nn \l__enumext_minipage_left_skip
1165                {
1166                  0.5875\box_dp:N \strutbox
1167                  - \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
```

```
1168              }
1169          \skip_set:Nn \l__enumext_minipage_right_skip
1170            {
1171              + \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
1172              + \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
1173            }
1174          \skip_set:Nn \l__enumext_minipage_after_skip
1175            {
1176              0.325\box_dp:N \strutbox
1177              + \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
1178            }
1179        }
1180      }
1181    }
```

(*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.

```
1182  \cs_new_protected:Nn \__enumext_zero_parsep:
1183    {
1184      \int_case:nn { \l__enumext_level_int }
1185        {
1186          { 2 }{
1187                  \skip_if_eq:nnF { \l__enumext_parsep_i_skip } { \c_zero_skip }
1188                    {
1189                      \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
1190                    }
1191                }
1192          { 3 }{
1193                  \skip_if_eq:nnF { \l__enumext_parsep_ii_skip } { \c_zero_skip }
1194                    {
1195                      \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
1196                    }
1197                }
1198          { 4 }{
1199                  \skip_if_eq:nnF { \l__enumext_parsep_iii_skip } { \c_zero_skip }
1200                    {
1201                      \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
1202                    }
1203                }
1204        }
1205    }
```

(*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*.

```
1206  \cs_new_protected:Nn \__enumext_mini_addvspace:
1207    {
1208      \__enumext_mini_set_vskip:
1209      \mode_if_vertical:T
1210        {
1211          \skip_add:Nn \l__enumext_minipage_left_skip
1212            {
1213              \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
1214            }
1215          \skip_add:Nn \l__enumext_minipage_after_skip
1216            {
1217              \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
1218            }
1219        }
1220      \par\nopagebreak
1221      \addvspace { \l__enumext_minipage_left_skip }
1222    }
```

(*End of definition for* __enumext_mini_addvspace:.)

11.21.2 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`.

```
1223 \cs_new_protected:Nn \__enumext_keyans_mini_set_vskip:
1224   {
1225     \skip_zero_new:N \l__enumext_minipage_after_skip
1226     \skip_zero_new:N \l__enumext_minipage_left_skip
1227     \skip_zero_new:N \l__enumext_minipage_right_skip
1228     \int_compare:nNnTF { \l__enumext_columns_v_int } > { 1 }
1229       {
1230         \skip_if_eq:nnTF { \l__enumext_topsep_v_skip } { \c_zero_skip }
1231           {
1232             \skip_set:Nn \l__enumext_minipage_left_skip { -0.25\box_dp:N \strutbox }
1233             \skip_set:Nn \l__enumext_minipage_right_skip { 0.705\box_dp:N \strutbox }
1234             \skip_set:Nn \l__enumext_minipage_after_skip { \box_dp:N \strutbox }
1235             \skip_if_eq:nnF { \l__enumext_parsep_i_skip } { \c_zero_skip }
1236               {
1237                 \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
1238               }
1239           }
1240           {
1241             \skip_set:Nn \l__enumext_minipage_left_skip
1242               {
1243                 \skip_use:N \l__enumext_topsep_v_skip
1244               }
1245             \skip_set:Nn \l__enumext_minipage_right_skip
1246               {
1247                 0.705\box_dp:N \strutbox
1248               }
1249             \skip_set:Nn \l__enumext_minipage_after_skip
1250               {
1251                 1.85\box_dp:N \strutbox + \l__enumext_topsep_v_skip
1252               }
1253           }
1254       }
1255       {
1256         \skip_if_eq:nnTF { \l__enumext_topsep_v_skip } { \c_zero_skip }
1257           {
1258             \skip_set:Nn \l__enumext_minipage_left_skip
1259               {
1260                 0.5\box_dp:N \strutbox
1261                 + \l__enumext_partopsep_v_skip
1262               }
1263             \skip_set:Nn \l__enumext_minipage_right_skip
1264               {
1265                 \l__enumext_partopsep_v_skip
1266               }
1267             \skip_set:Nn \l__enumext_minipage_after_skip { 1.6\box_dp:N \strutbox }
1268           }
1269           {
1270             \skip_set:Nn \l__enumext_minipage_left_skip
1271               {
1272                 0.5875\box_dp:N \strutbox - \l__enumext_partopsep_v_skip
1273               }
1274             \skip_set:Nn \l__enumext_minipage_right_skip
1275               {
1276                 \l__enumext_topsep_v_skip + \l__enumext_partopsep_v_skip
1277               }
1278             \skip_set:Nn \l__enumext_minipage_after_skip
1279               {
1280                 0.325\box_dp:N \strutbox + \l__enumext_topsep_v_skip
1281               }
1282           }
1283       }
1284   }
```

(*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`.

```
1285  \cs_new_protected:Nn \__enumext_keyans_mini_addvspace:
1286    {
1287      \__enumext_keyans_mini_set_vskip:
1288      \mode_if_vertical:T
1289        {
1290          \skip_add:Nn \l__enumext_minipage_left_skip
1291            {
1292              \l__enumext_partopsep_v_skip
1293            }
1294          \skip_add:Nn \l__enumext_minipage_after_skip
1295            {
1296              \l__enumext_partopsep_v_skip
1297            }
1298        }
1299      \par\nopagebreak
1300      \addvspace { \l__enumext_minipage_left_skip }
1301    }
```

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

11.21.3   Adjustment of vertical spaces for `minipage` in enumext* and keyans*

`\__enumext_mini_set_vskip_vii:`
`\__enumext_mini_set_vskip_viii:`

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

```
1302  \cs_new_protected:Nn \__enumext_mini_set_vskip_vii:
1303    {
1304      \skip_zero_new:N \l__enumext_minipage_left_skip
1305      \skip_gzero_new:N \g__enumext_minipage_right_skip
1306      \skip_gzero_new:N \g__enumext_minipage_after_skip
1307      \skip_if_eq:nnTF { \l__enumext_topsep_vii_skip } { \c_zero_skip }
1308        {
1309          \skip_set:Nn \l__enumext_minipage_left_skip { 0.5\box_dp:N \strutbox }
1310          \skip_gset:Nn \g__enumext_minipage_right_skip { 0.325\box_dp:N \strutbox }
1311        }
1312        {
1313          \skip_set:Nn \l__enumext_minipage_left_skip { 0.5875\box_dp:N \strutbox }
1314          \skip_gset:Nn \g__enumext_minipage_right_skip
1315            {
1316              \l__enumext_topsep_vii_skip
1317            }
1318          \skip_gset:Nn \g__enumext_minipage_after_skip
1319            {
1320              0.325\box_dp:N \strutbox + \l__enumext_topsep_vii_skip
1321            }
1322        }
1323    }
1324  \cs_new_protected:Nn \__enumext_mini_set_vskip_viii:
1325    {
1326      \skip_zero_new:N \l__enumext_minipage_after_skip
1327      \skip_zero_new:N \l__enumext_minipage_left_skip
1328      \skip_zero_new:N \l__enumext_minipage_right_skip
1329      \skip_if_eq:nnTF { \l__enumext_topsep_viii_skip } { \c_zero_skip }
1330        {
1331          \skip_set:Nn \l__enumext_minipage_left_skip
1332            {
1333              0.5\box_dp:N \strutbox
1334            }
1335          \skip_set:Nn \l__enumext_minipage_right_skip
1336            {
1337              \l__enumext_partopsep_viii_skip
1338            }
1339          \skip_set:Nn \l__enumext_minipage_after_skip
1340            {
1341              1.6\box_dp:N \strutbox
1342            }
1343        }
1344        {
```

```
1345        \skip_set:Nn \l__enumext_minipage_left_skip
1346          {
1347            0.5875\box_dp:N \strutbox
1348          }
1349        \skip_set:Nn \l__enumext_minipage_right_skip
1350          {
1351            \l__enumext_topsep_viii_skip
1352          }
1353        \skip_set:Nn \l__enumext_minipage_after_skip
1354          {
1355            0.325\box_dp:N \strutbox + \l__enumext_topsep_viii_skip
1356          }
1357        }
1358    }
```

(*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 mini-right 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.

```
1359  \cs_new_protected:Nn \__enumext_mini_addvspace_vii:
1360    {
1361      \__enumext_mini_set_vskip_vii:
1362      \par\nopagebreak
1363      \addvspace { \l__enumext_minipage_left_skip }
1364    }
1365  \cs_new_protected:Nn \__enumext_mini_addvspace_viii:
1366    {
1367      \__enumext_mini_set_vskip_viii:
1368      \par\nopagebreak
1369      \addvspace { \l__enumext_minipage_left_skip }
1370    }
```

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

11.21.4   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 argument* '*' 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.

```
1371  \NewDocumentCommand \miniright { s }
1372    {
1373      \int_compare:nNnT { \l__enumext_keyans_pic_level_int } = { 1 }
1374        {
1375          \msg_error:nnn { enumext } { wrong-miniright-place }
1376        }
1377      \int_compare:nNnT { \l__enumext_level_int } = { 0 }
1378        {
1379          \msg_error:nnn { enumext } { wrong-miniright-place }
1380        }
1381      \int_compare:nNnTF { \l__enumext_keyans_level_int } = { 1 }
1382        {
1383          \__enumext_keyans_mini_right_cmd:n {#1}
1384        }
1385        { \__enumext_mini_right_cmd:n {#1} }
1386    }
```

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

__enumext_mini_right_cmd:n

The function __enumext_mini_right_cmd:n takes as argument the *starred* '*' 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.

```
1387  \cs_new_protected:Npn \__enumext_mini_right_cmd:n #1
1388    {
1389      \dim_compare:nNnTF
1390        { \dim_use:c { l__enumext_minipage_right_ \__enumext_level: _dim } } > { \c_zero_dim }
1391        {
1392          \__enumext_multicols_stop:
1393          \end{__enumext_mini_env*}
1394          \hfill
1395          \begin{__enumext_mini_env*}
1396            { \dim_use:c { l__enumext_minipage_right_ \__enumext_level: _dim } }
1397            \par\addvspace { \l__enumext_minipage_right_skip }
1398            \bool_if:nF {#1}
1399              {
1400                \centering
1401              }
1402            \int_gzero:N \g__enumext_minipage_stat_int
1403        }
1404        { \msg_error:nnn { enumext } { wrong-miniright-use } }
1405    }
```

(*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* '`*`' 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.

```
1406  \cs_new_protected:Npn \__enumext_keyans_mini_right_cmd:n #1
1407    {
1408      \dim_compare:nNnTF { \l__enumext_minipage_right_v_dim } > { \c_zero_dim }
1409        {
1410          \__enumext_keyans_multicols_stop:
1411          \end{__enumext_mini_env*}
1412          \hfill
1413          \begin{__enumext_mini_env*}{ \l__enumext_minipage_right_v_dim }
1414            \par\addvspace { \l__enumext_minipage_right_skip }
1415            \bool_if:nF {#1}
1416              {
1417                \centering
1418              }
1419            \int_gzero:N \g__enumext_minipage_stat_int
1420        }
1421        { \msg_error:nnn { enumext } { wrong-miniright-use } }
1422    }
```

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

11.22   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 Define `above`, `above*`, `below` and `below*` keys for `enumext` and `keyans` environments.
above*
below
below*
```
1423  \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1424    {
1425      \keys_define:nn { enumext / #1 }
1426        {
1427          above  .skip_set:c = { l__enumext_vspace_above_#2_skip },
1428          above  .value_required:n = true,
1429          above* .code:n     = \bool_set_true:c { l__enumext_vspace_a_star_#2_bool }
1430                               \keys_set:nn { enumext / #1 } { above = {##1} },
1431          above* .value_required:n = true,
1432          below  .skip_set:c = { l__enumext_vspace_below_#2_skip },
1433          below  .value_required:n = true,
1434          below* .code:n     = \bool_set_true:c { l__enumext_vspace_b_star_#2_bool }
1435                               \keys_set:nn { enumext / #1 } { below = {##1} },
1436          below* .value_required:n = true,
```

```
1437           }
1438     }
1439   \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }
```

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

11.22.1   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.

```
1440   \cs_new_protected:Nn \__enumext_vspace_above:
1441     {
1442       \skip_if_eq:nnF
1443         { \skip_use:c { l__enumext_vspace_above_ \__enumext_level: _skip } } { \c_zero_skip }
1444         {
1445           \bool_if:cTF { l__enumext_vspace_a_star_ \__enumext_level: _bool }
1446             {
1447               \vspace*{ \skip_use:c { l__enumext_vspace_above_ \__enumext_level: _skip } }
1448             }
1449             {
1450               \vspace { \skip_use:c { l__enumext_vspace_above_ \__enumext_level: _skip } }
1451             }
1452         }
1453     }
```

(*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.

```
1454   \cs_new_protected:Nn \__enumext_vspace_below:
1455     {
1456       \skip_if_eq:nnF
1457         { \skip_use:c { l__enumext_vspace_below_ \__enumext_level: _skip } } { \c_zero_skip }
1458         {
1459           \bool_if:cTF { l__enumext_vspace_b_star_ \__enumext_level: _bool }
1460             {
1461               \vspace*{ \skip_use:c { l__enumext_vspace_below_ \__enumext_level: _skip } }
1462             }
1463             {
1464               \vspace { \skip_use:c { l__enumext_vspace_below_ \__enumext_level: _skip } }
1465             }
1466         }
1467     }
```

(*End of definition for __enumext_vspace_below:.*)

11.22.2   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.

```
1468   \cs_new_protected:Nn \__enumext_vspace_above_v:
1469     {
1470       \skip_if_eq:nnF { \l__enumext_vspace_above_v_skip } { \c_zero_skip }
1471         {
1472           \bool_if:NTF \l__enumext_vspace_a_star_v_bool
1473             {
1474               \vspace*{ \l__enumext_vspace_above_v_skip }
1475             }
1476             { \vspace { \l__enumext_vspace_above_v_skip } }
1477         }
1478     }
```

(*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.

```
1479   \cs_new_protected:Nn \__enumext_vspace_below_v:
1480     {
1481       \skip_if_eq:nnF { \l__enumext_vspace_below_v_skip } { \c_zero_skip }
1482         {
1483           \bool_if:NTF \l__enumext_vspace_b_star_v_bool
1484             {
1485               \vspace*{ \l__enumext_vspace_below_v_skip }
```

```
1486              }
1487            { \vspace { \l__enumext_vspace_below_v_skip } }
1488          }
1489      }
```

(*End of definition for __enumext_vspace_below_v:.*)

11.22.3   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.

```
1490  \cs_new_protected:Nn \__enumext_vspace_above_vii:
1491    {
1492      \skip_if_eq:nnF { \l__enumext_vspace_above_vii_skip } { \c_zero_skip }
1493        {
1494          \bool_if:NTF \l__enumext_vspace_a_star_vii_bool
1495            {
1496              \vspace*{ \l__enumext_vspace_above_vii_skip }
1497            }
1498            { \vspace { \l__enumext_vspace_above_vii_skip } }
1499        }
1500    }
1501  \cs_new_protected:Nn \__enumext_vspace_above_viii:
1502    {
1503      \skip_if_eq:nnF { \l__enumext_vspace_above_viii_skip } { \c_zero_skip }
1504        {
1505          \bool_if:NTF \l__enumext_vspace_a_star_viii_bool
1506            {
1507              \vspace*{ \l__enumext_vspace_above_viii_skip }
1508            }
1509            { \vspace { \l__enumext_vspace_above_viii_skip } }
1510        }
1511    }
```

(*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.

```
1512  \cs_new_protected:Nn \__enumext_vspace_below_vii:
1513    {
1514      \skip_if_eq:nnF { \l__enumext_vspace_below_vii_skip } { \c_zero_skip }
1515        {
1516          \bool_if:NTF \l__enumext_vspace_b_star_vii_bool
1517            {
1518              \vspace*{ \l__enumext_vspace_below_vii_skip }
1519            }
1520            { \vspace { \l__enumext_vspace_below_vii_skip } }
1521        }
1522    }
1523  \cs_new_protected:Nn \__enumext_vspace_below_viii:
1524    {
1525      \skip_if_eq:nnF { \l__enumext_vspace_below_viii_skip } { \c_zero_skip }
1526        {
1527          \bool_if:NTF \l__enumext_vspace_b_star_viii_bool
1528            {
1529              \vspace*{ \l__enumext_vspace_below_viii_skip }
1530            }
1531            { \vspace { \l__enumext_vspace_below_viii_skip } }
1532        }
1533    }
```

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

11.23   Setting series, resume and resume* keys

The series key is responsible for the whole process of the resume and resume* keys. The idea behind this is to be able to absorb the ⟨keys⟩ passed to the optional argument of the *"first level"* of the environments enumext and enumext*, but, discarding some specific ⟨keys⟩. This implementation is adapted directly from the code provided by Jonathan P. Spratte (@Skillmon) in chat-TeX-SX

series
resume
resume*

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

```
1534 \cs_set_protected:Npn \__enumext_tmp:n #1
1535   {
1536     \keys_define:nn { enumext / #1 }
1537       {
1538         series  .str_set:N = \l__enumext_series_str,
1539         series  .value_required:n = true,
1540         resume  .code:n = \__enumext_resume_series:n {##1},
1541         resume* .code:n = \__enumext_resume_starred:,
1542         resume* .value_forbidden:n = true,
1543       }
1544   }
1545 \clist_map_inline:nn { level-1, enumext* } { \__enumext_tmp:n {#1} }
```

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

11.23.1 Internal functions for series key

__enumext_filter_series:n
__enumext_filter_series_key:n
__enumext_filter_series_pair:nn

The function `\__enumext_filter_series:n` will be in charge of filtering the ⟨*keys*⟩ we want to store where {`#1`} represents the optional value passed to the environment.

```
1546 \cs_new:Npn \__enumext_filter_series:n #1
1547   {
1548     \use:e
1549       {
1550         \keyval_parse:NNn
1551           \__enumext_filter_series_key:n
1552           \__enumext_filter_series_pair:nn {#1}
1553       }
1554   }
```

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

```
1555 \cs_new:Npn \__enumext_filter_series_key:n #1
1556   {
1557     \str_case:nnF {#1}
1558       {
1559         { resume   } {}
1560         { resume*  } {}
1561         { base-fix } {}
1562       }
1563       { , { \exp_not:n {#1} } }
1564   }
```

The function `\__enumext_filter_series_pair:nn` will be responsible for filtering the ⟨*keys*⟩ that are passed *"with value"* by excluding the `series`, `resume`, `start`, `save-ans` and `save-key` keys.

```
1565 \cs_new:Npn \__enumext_filter_series_pair:nn #1#2
1566   {
1567     \str_case:nnF {#1}
1568       {
1569         { series } {}
1570         { resume } {}
1571         { start  } {}
1572         { save-ans } {}
1573         { save-key } {}
1574       }
1575       { , { \exp_not:n {#1} } = { \exp_not:n {#2} } }
1576   }
```

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

__enumext_parse_series:n
__enumext_resume_last:n

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

⚽ The value of boolean variable `\l__enumext_resume_active_bool` is set to true by the function `\__enumext_-resume_counter:n` which is used by the keys `resume` and `resume*`, in this case we must Make sure it is set to false so that it does not overwrite the default filtered ⟨*keys*⟩. This function is passed to the function `\__enumext_parse_-keys:n` in the `enumext` environment definition (§11.35) and to the function `\__enumext_parse_keys_vii:n` in the `enumext*` environment definition (§11.39).

```
1577 \cs_new_protected:Npn \__enumext_parse_series:n #1
```

```
1578  {
1579    \str_if_empty:NTF \l__enumext_series_str
1580      {
1581        \bool_if:NF \l__enumext_resume_active_bool
1582          {
1583            \__enumext_resume_last:n {#1}
1584          }
1585      }
1586      {
1587        \tl_gclear_new:c { g__enumext_series_ \l__enumext_series_str _tl }
1588        \tl_gset:ce { g__enumext_series_ \l__enumext_series_str _tl }
1589          { \__enumext_filter_series:n {#1} }
1590        \int_if_exist:cF { g__enumext_series_ \l__enumext_series_str _int }
1591          {
1592            \int_new:c { g__enumext_series_ \l__enumext_series_str _int }
1593          }
1594      }
1595  }
```

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

```
1596  \cs_new_protected:Npn \__enumext_resume_last:n #1
1597    {
1598      \bool_if:NT \l__enumext_standar_first_bool
1599        {
1600          \tl_gclear:N \g__enumext_standar_series_tl
1601          \tl_gset:Ne \g__enumext_standar_series_tl { \__enumext_filter_series:n {#1} }
1602        }
1603      \bool_if:NT \l__enumext_starred_first_bool
1604        {
1605          \tl_gclear:N \g__enumext_starred_series_tl
1606          \tl_gset:Ne \g__enumext_starred_series_tl { \__enumext_filter_series:n {#1} }
1607        }
1608    }
```

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

11.23.2   Internal function to save counter value

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

💣 The variables \l__enumext_series_str and \l__enumext__resume_name_tl contain the same {⟨*series name*⟩} but are executed at different moments, the integer variable with \l__enumext_series_str sets the value when execute series={⟨*series name*⟩} and the integer variable with \l__enumext__resume_name_tl sets the subsequent values when use resume={⟨*series name*⟩}. This function is passed to the enumext environment definition (§11.35) and the enumext* environment definition (§11.39).

```
1609  \cs_new_protected:Nn \__enumext_resume_save_counter:
1610    {
1611      \bool_if:NT \g__enumext_standar_bool
1612        {
1613          \tl_if_empty:NF \l__enumext_series_str
1614            {
1615              \int_gset_eq:cN
1616                { g__enumext_series_ \l__enumext_series_str _int } \value{enumXi}
1617            }
1618          \tl_if_empty:NTF \l__enumext_resume_name_tl
1619            {
1620              \str_if_empty:NT \l__enumext_series_str
1621                {
1622                  \int_gset_eq:NN \g__enumext_resume_int \value{enumXi}
1623                }
1624            }
1625            {
1626              \int_if_exist:cT { g__enumext_series_ \l__enumext_resume_name_tl _int }
1627                {
```

```
1628                    \int_gset_eq:cN
1629                      { g__enumext_series_ \l__enumext_resume_name_tl _int } \value{enumXi}
1630                  }
1631              }
1632          \int_if_exist:cT { g__enumext_resume_ \l__enumext_store_name_tl _int }
1633            {
1634              \int_gset_eq:cN
1635                { g__enumext_resume_ \l__enumext_store_name_tl _int } \value{enumXi}
1636            }
1637        }
1638      \bool_if:NT \g__enumext_starred_bool
1639        {
1640          \tl_if_empty:NF \l__enumext_series_str
1641            {
1642              \int_gset_eq:cN
1643                { g__enumext_series_ \l__enumext_series_str _int } \value{enumXvii}
1644            }
1645          \tl_if_empty:NTF \l__enumext_resume_name_tl
1646            {
1647              \str_if_empty:NT \l__enumext_series_str
1648                {
1649                  \int_gset_eq:NN \g__enumext_resume_vii_int \value{enumXvii}
1650                }
1651            }
1652            {
1653              \int_if_exist:cT { g__enumext_series_ \l__enumext_resume_name_tl _int }
1654                {
1655                  \int_gset_eq:cN
1656                    { g__enumext_series_ \l__enumext_resume_name_tl _int } \value{enumXvii}
1657                }
1658            }
1659          \int_if_exist:cT { g__enumext_resume_ \l__enumext_store_name_tl _int }
1660            {
1661              \int_gset_eq:cN
1662                { g__enumext_resume_ \l__enumext_store_name_tl _int } \value{enumXvii}
1663            }
1664        }
1665    }
```

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

11.23.3   Internal functions for resume key

`\__enumext_resume_series:n`   The function `\__enumext_resume_series:n` will handle the argument passed to the `resume` key in `enumext` and `enumext*` environments. If the key is passed *without value* the function `\__enumext_-resume_counter:` is executed which will set the counter according to the numbering of the last `enumext` or `enumext*` environments in which `series={⟨series name⟩}` key is not present, if the `save-ans` key is active it will set the counter according to the value of the integer variable created by that key, otherwise it will verify that the `\g__enumext_series_⟨series name⟩_tl` variable set by the `series` key exists, if so it will pass these keys to the *first level* of the environment, otherwise it will return an error.

```
1666  \cs_new_protected:Npn \__enumext_resume_series:n #1
1667    {
1668      \tl_if_empty:nTF {#1}
1669        {
1670          \__enumext_resume_counter:n { }
1671        }
1672        {
1673          \tl_if_exist:cTF { g__enumext_series_ \tl_to_str:n {#1} _tl }
1674            {
1675              \__enumext_resume_counter:n {#1}
1676              \bool_if:NT \g__enumext_standar_bool
1677                {
1678                  \keys_set:nv { enumext / level-1 }
1679                    { g__enumext_series_ \tl_to_str:n {#1} _tl }
1680                }
1681              \bool_if:NT \g__enumext_starred_bool
1682                {
1683                  \keys_set:nv { enumext / enumext* }
1684                    { g__enumext_series_ \tl_to_str:n {#1} _tl }
1685                }
1686            }
1687            {
```

```
1688            \bool_if:NT \g__enumext_standar_bool
1689                {
1690                   \msg_error:nnn { enumext } { unknown-series } {#1}
1691                }
1692            \bool_if:NT \g__enumext_starred_bool
1693                {
1694                   \msg_error:nnn { enumext } { unknown-series } {#1}
1695                }
1696         }
1697       }
1698    }
```

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

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

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

```
1699  \cs_new_protected:Npn \__enumext_resume_counter:n #1
1700    {
1701       \bool_set_true:N \l__enumext_resume_active_bool
1702       \tl_set:Nn \l__enumext_resume_name_tl {#1}
1703       \tl_if_empty:NTF \l__enumext_resume_name_tl
1704         {
1705            \__enumext_resume_counter:
1706         }
1707         {
1708            \__enumext_resume_counter_series:
1709         }
1710       \__enumext_resume_counter_save_ans:
1711    }
```

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

```
1712  \cs_new_protected:Nn \__enumext_resume_counter:
1713    {
1714       \bool_if:NT \g__enumext_standar_bool
1715         {
1716            \int_gincr:N \g__enumext_resume_int
1717            \int_set_eq:NN \l__enumext_start_i_int \g__enumext_resume_int
1718         }
1719       \bool_if:NT \g__enumext_starred_bool
1720         {
1721            \int_gincr:N \g__enumext_resume_vii_int
1722            \int_set_eq:NN \l__enumext_start_vii_int \g__enumext_resume_vii_int
1723         }
1724    }
```

The function `\__enumext_resume_counter_series:` will be executed when the resume={⟨*series name*⟩} key is active, setting the counters for the *"first level"* of the environments according to the value of the integer variables created by the series key.

```
1725  \cs_new_protected:Nn \__enumext_resume_counter_series:
1726    {
1727       \bool_if:NT \g__enumext_standar_bool
1728         {
1729            \int_set:Nn \l__enumext_start_i_int
1730              {
1731                 \int_use:c { g__enumext_series_ \l__enumext_resume_name_tl _int } + 1
1732              }
1733         }
1734       \bool_if:NT \g__enumext_starred_bool
1735         {
1736            \int_set:Nn \l__enumext_start_vii_int
1737              {
1738                 \int_use:c { g__enumext_series_ \l__enumext_resume_name_tl _int } + 1
1739              }
1740         }
1741    }
```

The function __enumext_resume_counter_save_ans: will be executed when the save-ans key is active along with the resume key, setting the counters for the *"first level"* of the environments according to the value of the integer variables created by the save-ans key.

```
1742 \cs_new_protected:Nn \__enumext_resume_counter_save_ans:
1743   {
1744     \bool_lazy_and:nnT
1745       { \bool_if_p:N \l__enumext_standar_first_bool }
1746       { \bool_if_p:N \l__enumext_store_active_bool }
1747       {
1748         \int_set:Nn \l__enumext_start_i_int
1749           {
1750             \int_use:c { g__enumext_resume_ \l__enumext_store_name_tl _int } + 1
1751           }
1752       }
1753     \bool_lazy_and:nnT
1754       { \bool_if_p:N \l__enumext_starred_first_bool }
1755       { \bool_if_p:N \l__enumext_store_active_bool }
1756       {
1757         \int_set:Nn \l__enumext_start_vii_int
1758           {
1759             \int_use:c { g__enumext_resume_ \l__enumext_store_name_tl _int } + 1
1760           }
1761       }
1762   }
```

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

11.23.4   Internal function for resume* key

__enumext_resume_starred: The function __enumext_resume_starred: will handle the resume* key in the enumext and enumext* environments. This function will execute the filtered ⟨*keys*⟩ in the last one and will continue with the numbering according to the last execution of the environment enumext or enumext* in which the keys resume={⟨*series name*⟩} or series={⟨*series name*⟩} were not active.

```
1763 \cs_new_protected:Nn \__enumext_resume_starred:
1764   {
1765     \bool_if:NT \g__enumext_standar_bool
1766       {
1767         \tl_if_empty:NF \g__enumext_standar_series_tl
1768           {
1769             \__enumext_resume_counter:n { }
1770             \keys_set:nV { enumext / level-1 } \g__enumext_standar_series_tl
1771           }
1772       }
1773     \bool_if:NT \g__enumext_starred_bool
1774       {
1775         \tl_if_empty:NF \g__enumext_starred_series_tl
1776           {
1777             \__enumext_resume_counter:n { }
1778             \keys_set:nV { enumext / enumext* } \g__enumext_starred_series_tl
1779           }
1780       }
1781   }
```

(*End of definition for* __enumext_resume_starred:.)

11.24   Setting save-ans, check-ans and no-store keys

The key save-ans is directly associated with the keys check-ans, no-store, resume and resume*, this will activate the entire *"storage system"* in the enumext package.

11.24.1   Setting save-ans key

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

```
1782 \cs_set_protected:Npn \__enumext_tmp:n #1
1783   {
1784     \keys_define:nn { enumext / #1 }
1785       {
1786         save-ans .code:n = \__enumext_storing_set:n {##1},
1787         save-ans .value_required:n = true,
1788       }
1789   }
1790 \clist_map_inline:nn { level-1, enumext* } { \__enumext_tmp:n {#1} }
```

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

11.24.2   Internal functions for save-ans key

__enumext_start_save_ans_msg:
__enumext_stop_save_ans_msg:

The functions __enumext_start_save_ans_msg: and __enumext_stop_save_ans_msg: will display in the terminal and .log file the environment in which the save-ans key was executed along with the line at the beginning and end of it. The function __enumext_start_save_ans_msg: will be passed to __enumext_storing_set:n and the function __enumext_stop_save_ans_msg: will be passed to the function __enumext_execute_after_env:.

```
1791 \cs_new_protected:Nn \__enumext_start_save_ans_msg:
1792   {
1793     \msg_term:nnVV { enumext } { save-ans-log }
1794       \g__enumext_envir_name_tl \l__enumext_store_name_tl
1795   }
1796 \cs_new_protected:Nn \__enumext_stop_save_ans_msg:
1797   {
1798     \msg_term:nnVV { enumext } { save-ans-log-hook }
1799       \g__enumext_envir_name_tl \g__enumext_store_name_tl
1800   }
```

(*End of definition for* __enumext_start_save_ans_msg: *and* __enumext_stop_save_ans_msg:.)

__enumext_storing_set:n
__enumext_storing_exec:

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

```
1801 \cs_new_protected:Npn \__enumext_storing_set:n #1
1802   {
1803     \tl_set:Ne \l__enumext_store_name_tl {#1}
1804     \tl_if_empty:NTF \l__enumext_store_name_tl
1805       {
1806         \bool_lazy_or:nnT
1807           { \l__enumext_standar_first_bool } { \l__enumext_starred_first_bool }
1808           {
1809             \msg_error:nnV { enumext } { save-ans-empty } \g__enumext_envir_name_tl
1810           }
1811       }
1812       {
1813         \bool_lazy_or:nnT
1814           { \l__enumext_standar_first_bool } { \l__enumext_starred_first_bool }
1815           {
1816             \__enumext_start_save_ans_msg:
1817             \__enumext_storing_exec:
1818           }
1819       }
1820   }
```

The function __enumext_storing_exec: will set to true the variable \l__enumext_store_active_-bool which activates the use of the \anskey command and the keyans, keyans* and keyanspic environments and will set to true the variable \l__enumext_check_answers_bool used for checking answers by the check-ans and no-store keys, copy {⟨*store name*⟩} into the global variable \g__enumext_-store_name_tl and execute the function __enumext_anskey_env_make:V creating the environment anskey* (§11.28). The ⟨*prop list*⟩ \g__enumext_series_⟨*store name*⟩_prop and the ⟨*sequence*⟩ \g__-enumext_series_⟨*store name*⟩_seq will be created globally to *"store content"* in case they do not exist together with the integer variable \g__enumext_series_⟨*store name*⟩_int used by the keys resume and resume*.

```
1821 \cs_new_protected:Nn \__enumext_storing_exec:
1822   {
1823     \bool_set_true:N \l__enumext_store_active_bool
1824     \bool_set_true:N \l__enumext_check_answers_bool
1825     \tl_gset:NV \g__enumext_store_name_tl \l__enumext_store_name_tl
1826     \__enumext_anskey_env_make:V \l__enumext_store_name_tl
1827     \prop_if_exist:cF { g__enumext_ \l__enumext_store_name_tl _prop }
1828       {
1829         \msg_log:nnV { enumext } { store-prop } \l__enumext_store_name_tl
1830         \prop_new:c { g__enumext_ \l__enumext_store_name_tl _prop }
1831       }
1832     \seq_if_exist:cF { g__enumext_ \l__enumext_store_name_tl _seq }
1833       {
1834         \msg_log:nnV { enumext } { store-seq } \l__enumext_store_name_tl
1835         \seq_new:c { g__enumext_ \l__enumext_store_name_tl _seq }
```

```
1836            }
1837        \int_if_exist:cF { g__enumext_resume_ \l__enumext_store_name_tl _int }
1838          {
1839            \msg_log:nnV { enumext } { store-int } \l__enumext_store_name_tl
1840            \int_new:c { g__enumext_resume_ \l__enumext_store_name_tl _int }
1841          }
1842      }
```

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

11.24.3   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_item_number_int must match the integer variable \g__enumext_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.

11.24.4   Setting check-ans and no-store keys

check-ans
no-store

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

```
1843 \cs_set_protected:Npn \__enumext_tmp:n #1
1844   {
1845     \keys_define:nn { enumext / #1 }
1846       {
1847         check-ans .bool_set:N = \l__enumext_check_ans_key_bool,
1848         check-ans .initial:n  = false,
1849         check-ans .value_required:n = true,
1850         no-store  .code:n = {
1851                             \bool_set_false:N \l__enumext_check_answers_bool
1852                             \bool_set_false:N \l__enumext_check_ans_key_bool
1853                           },
1854         no-store  .value_forbidden:n = true,
1855       }
1856   }
1857 \clist_map_inline:nn
1858   {
1859     level-1, level-2, level-3, level-4, enumext*
1860   }
1861   { \__enumext_tmp:n {#1} }
```

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

11.24.5   Set-up check answer mechanism

__enumext_check_ans_active:
__enumext_check_ans_level:

The function __enumext_check_ans_active: will first check the state of the variable \l__enumext_-store_name_tl, that is, the save-ans key is active, if so it will check the state of the variable \l__enumext_check_answers_bool handled by the key no-store and will execute the function __enumext_check_ans_level: only if *"true"*, i.e. the key no-store is not active.

```
1862 \cs_new_protected:Nn \__enumext_check_ans_active:
1863   {
1864     \tl_if_empty:NF \l__enumext_store_name_tl
1865       {
1866         \bool_if:NT \l__enumext_check_answers_bool
1867           {
1868             \__enumext_check_ans_level:
1869           }
1870       }
1871   }
```

The function __enumext_check_ans_level: will decrement by *"one"* the value of the variable \g__-enumext_item_number_int which keeps track of the executions of \item and \item* for each level of nesting of the environment enumext, taking into account whether it is nested within enumext* or the opposite and set \l__enumext_item_number_bool to *"false"* .

```
1872 \cs_new_protected:Nn \__enumext_check_ans_level:
1873   {
1874     \int_case:nn { \l__enumext_level_int }
1875       {
1876         { 1 }{
1877             \bool_lazy_all:nT
1878               {
1879                 { \bool_if_p:N \g__enumext_starred_bool }
1880                 { \int_compare_p:nNn { \l__enumext_level_h_int } = { 1 } }
1881               }
1882               {
1883                 \int_gdecr:N \g__enumext_item_number_int
1884                 \bool_set_false:N \l__enumext_item_number_bool
1885               }
1886           }
1887         { 2 }{
1888             \int_gdecr:N \g__enumext_item_number_int
1889             \bool_set_false:N \l__enumext_item_number_bool
1890           }
1891         { 3 }{
1892             \int_gdecr:N \g__enumext_item_number_int
1893             \bool_set_false:N \l__enumext_item_number_bool
1894           }
1895         { 4 }{
1896             \int_gdecr:N \g__enumext_item_number_int
1897             \bool_set_false:N \l__enumext_item_number_bool
1898           }
1899       }
```

We should only execute this if enumext* is nested in the first level of enumext, for the rest of the cases the value of \g__enumext_item_number_int is already decreased.

```
1900     \int_case:nn { \l__enumext_level_h_int }
1901       {
1902         { 1 }{
1903             \bool_lazy_all:nT
1904               {
1905                 { \bool_if_p:N \g__enumext_standar_bool }
1906                 { \int_compare_p:nNn { \l__enumext_level_int } = { 1 } }
1907               }
1908               {
1909                 \int_gdecr:N \g__enumext_item_number_int
1910                 \bool_set_false:N \l__enumext_item_number_bool
1911               }
1912           }
1913       }
1914   }
```

(*End of definition for* __enumext_check_ans_active: *and* __enumext_check_ans_level:*.*)

__enumext_check_ans_key_hook: The function __enumext_check_ans_key_hook: will *export* the status of the local variable \l__-enumext_check_ans_key_bool to the global variable \g__enumext_check_ans_key_bool only if the key check-ans is active.

```
1915  \cs_new_protected:Nn \__enumext_check_ans_key_hook:
1916    {
1917      \bool_lazy_and:nnT
1918        { \bool_if_p:N \l__enumext_check_ans_key_bool }
1919        { \bool_if_p:N \g__enumext_standar_bool }
1920        {
1921          \bool_gset_true:N \g__enumext_check_ans_key_bool
1922        }
1923      \bool_lazy_and:nnT
1924        { \bool_if_p:N \l__enumext_check_ans_key_bool }
1925        { \bool_if_p:N \g__enumext_starred_bool }
1926        {
1927          \bool_gset_true:N \g__enumext_check_ans_key_bool
1928        }
1929    }
```

(*End of definition for* __enumext_check_ans_key_hook:.)

__enumext_item_answer_diff: The function __enumext_item_answer_diff: will set the value of the variable \g__enumext_item_-answer_diff_int which is used by the functions __enumext_check_ans_show: for the key save-ans and by the function __enumext_check_ans_log: by the internal *"check answer"* mechanism. This function will be passed to the function __enumext_execute_after_env:.

```
1930  \cs_new_protected:Nn \__enumext_item_answer_diff:
1931    {
1932      \int_gset:Nn \g__enumext_item_answer_diff_int
1933        {
1934          \int_sign:n { \g__enumext_item_number_int - \g__enumext_item_anskey_int }
1935        }
1936    }
```

(*End of definition for* __enumext_item_answer_diff:.)

__enumext_check_ans_show: The function __enumext_check_ans_show: will be executed within the function __enumext_-execute_after_env: when the key check-ans is active, that is, when \g__enumext_check_ans_-key_bool is *"true"* and will return the appropriate message according to the value of \g__enumext_-item_answer_diff_int set by the function __enumext_item_answer_diff:.

__enumext_check_ans_msg_less:
__enumext_check_ans_msg_same_ok:
__enumext_check_ans_msg_greater:

```
1937  \cs_new_protected:Nn \__enumext_check_ans_show:
1938    {
1939      \int_case:nn { \g__enumext_item_answer_diff_int }
1940        {
1941          { -1 }{ \__enumext_check_ans_msg_less:      }
1942          {  0 }{ \__enumext_check_ans_msg_same_ok: }
1943          {  1 }{ \__enumext_check_ans_msg_greater: }
1944        }
1945    }
1946  \cs_new_protected:Nn \__enumext_check_ans_msg_less:
1947    {
1948      \msg_warning:nneee { enumext } { item-less-answer } { \g__enumext_store_name_tl }
1949        { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
1950    }
1951  \cs_new_protected:Nn \__enumext_check_ans_msg_same_ok:
1952    {
1953      \msg_term:nneee { enumext } { items-same-answer } { \g__enumext_store_name_tl }
1954        { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
1955    }
1956  \cs_new_protected:Nn \__enumext_check_ans_msg_greater:
1957    {
1958      \msg_warning:nneee { enumext } { item-greater-answer } { \g__enumext_store_name_tl }
1959        { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
1960    }
```

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

__enumext_check_ans_log: The function __enumext_check_ans_log: will be executed within the function __enumext_-execute_after_env: when the key check-ans is not active, that is, when \g__enumext_check_-ans_key_bool is *"false"* and write in the log the appropriate message according to the value of \g__-enumext_item_answer_diff_int set by the function __enumext_item_answer_diff:.

__enumext_check_ans_log_msg_less:
__enumext_check_ans_log_msg_same_ok:
__enumext_check_ans_log_msg_greater:

```
1961  \cs_new_protected:Nn \__enumext_check_ans_log:
1962    {
1963      \int_case:nn { \g__enumext_item_answer_diff_int }
1964        {
1965          { -1 }{ \__enumext_check_ans_log_msg_less:     }
1966          {  0 }{ \__enumext_check_ans_log_msg_same_ok: }
1967          {  1 }{ \__enumext_check_ans_log_msg_greater: }
1968        }
1969    }
1970  \cs_new_protected:Nn \__enumext_check_ans_log_msg_less:
1971    {
1972      \msg_log:nneee { enumext } { item-less-answer } { \g__enumext_store_name_tl }
1973        { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
1974    }
1975  \cs_new_protected:Nn \__enumext_check_ans_log_msg_same_ok:
1976    {
1977      \msg_log:nneee { enumext } { items-same-answer }  { \g__enumext_store_name_tl }
1978        { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
1979    }
1980  \cs_new_protected:Nn \__enumext_check_ans_log_msg_greater:
1981    {
1982      \msg_log:nneee { enumext } { item-greater-answer } { \g__enumext_store_name_tl }
1983        { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
1984    }
```

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

11.24.6   Check for \item* and \anspic* commands

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

```
1985  \cs_new_protected:Npn \__enumext_check_starred_cmd:n #1
1986    {
1987      \int_compare:nNnT
1988        { \g__enumext_check_starred_cmd_int } = { 0 }
1989        {
1990          \msg_warning:nnnV
1991            { enumext } { missing-starred }{ #1 } \l__enumext_check_start_line_env_tl
1992        }
1993      \int_compare:nNnT
1994        { \g__enumext_check_starred_cmd_int } > { 1 }
1995        {
1996          \msg_warning:nnnV
1997            { enumext } { many-starred }{ #1 } \l__enumext_check_start_line_env_tl
1998        }
1999      \int_gzero:N \g__enumext_check_starred_cmd_int
2000      \tl_clear:N \l__enumext_check_start_line_env_tl
2001    }
```

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

11.25   Executing anskey*, check-ans and write .log

__enumext_execute_after_env: The __enumext_execute_after_env: function will first return the appropriate message for the end
of the environment in which the save-ans key is being executed, then call the __enumext_item_-
answer_diff: function and then will write the values of the global variables used to the .log file. If the key
check-ans is active it will execute the function __enumext_check_ans_show: and show the result in
the terminal, otherwise it will execute the function __enumext_check_ans_log: and write the results in
the .log file, undefine the environment anskey* (§11.28) through the function __enumext_undefine_-
anskey_env: and finally we execute the function __enumext_reset_global_vars: returning the
used variables to their original state.

```
2002  \cs_new_protected:Nn \__enumext_execute_after_env:
2003    {
2004      \int_compare:nNnT { \l__enumext_level_int } = { 0 }
2005        {
2006          \tl_if_empty:NF \g__enumext_store_name_tl
2007            {
2008              \__enumext_stop_save_ans_msg:
2009              \__enumext_item_answer_diff:
2010              \__enumext_log_global_vars:
2011              \__enumext_log_answer_vars:
```

```
2012                \bool_if:NTF \g__enumext_check_ans_key_bool
2013                  {
2014                    \__enumext_check_ans_show:
2015                  }
2016                  { \__enumext_check_ans_log: }
2017              \__enumext_undefine_anskey_env:
2018            }
2019          \__enumext_reset_global_vars:
2020        }
2021    }
```

💣 This function is passed to the function __enumext_after_env:nn for the environments enumext (§11.35) and enumext* (§11.39) and it is executed only when the environments are not nested or at some level of these..

(*End of definition for* __enumext_execute_after_env:.)

11.26   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*.

```
2022 \cs_set_protected:Npn \__enumext_tmp:n #1
2023   {
2024      \keys_define:nn { enumext / #1 }
2025        {
2026           wrap-ans    .cs_set_protected:Np = \__enumext_anskey_wrapper:n ##1,
2027           wrap-ans    .initial:n = \fbox{##1},
2028           wrap-ans    .value_required:n = true,
2029           wrap-opt    .cs_set_protected:Np = \__enumext_keyans_wrapper_opt:n ##1,
2030           wrap-opt    .initial:n = [{##1}],
2031           wrap-opt    .value_required:n = true,
2032           save-sep    .tl_set:N  = \l__enumext_store_keyans_item_opt_sep_tl,
2033           save-sep    .initial:n = {, ~ },
2034           save-sep    .value_required:n = true,
2035           mark-ans    .tl_set:N  = \l__enumext_mark_answer_sym_tl,
2036           mark-ans    .initial:n = \textasteriskcentered,
2037           mark-ans    .value_required:n = true,
2038           mark-pos    .choice:,
2039           mark-pos / left     .code:n = \str_set:Nn \l__enumext_mark_position_str { l },
2040           mark-pos / right    .code:n = \str_set:Nn \l__enumext_mark_position_str { r },
2041           mark-pos / unknown .code:n =
2042                                  \msg_error:nneee { enumext } { unknown-choice }
2043                                    { mark-pos } { left, ~ right } { \exp_not:n {##1} },
2044           mark-pos    .initial:n = right,
2045           mark-pos    .value_required:n = true,
2046           show-ans    .bool_set:N = \l__enumext_show_answer_bool,
2047           show-ans    .initial:n  = false,
2048           show-ans    .value_required:n = true,
2049           show-pos    .bool_set:N = \l__enumext_show_position_bool,
2050           show-pos    .initial:n  = false,
2051           show-pos    .value_required:n = true,
2052           mark-ref    .tl_set:N   = \l__enumext_mark_ref_sym_tl,
2053           mark-ref    .initial:n   = \textasteriskcentered,
2054           mark-ref    .value_required:n = true,
2055           save-ref    .bool_set:N = \l__enumext_store_ref_key_bool,
2056           save-ref    .initial:n  = false,
2057           save-ref    .value_required:n = true,
2058        }
2059   }
2060 \clist_map_inline:nn { level-1, enumext* } { \__enumext_tmp:n {#1} }
```

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

mark-pos
show-ans
show-pos

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

```
2061 \cs_set_protected:Npn \__enumext_tmp:n #1
2062   {
2063      \keys_define:nn { enumext / #1 }
2064        {
2065           mark-pos .choice:,
2066           mark-pos / left  .code:n = \str_set:Nn \l__enumext_mark_position_str { l },
2067           mark-pos / right .code:n = \str_set:Nn \l__enumext_mark_position_str { r },
```

```
2068        mark-pos .initial:n = right,
2069        mark-pos .value_required:n  = true,
2070        show-ans .bool_set:N = \l__enumext_show_answer_bool,
2071        show-ans .initial:n  = false,
2072        show-ans .value_required:n = true,
2073        show-pos .bool_set:N = \l__enumext_show_position_bool,
2074        show-pos .initial:n  = false,
2075        show-pos .value_required:n = true,
2076      }
2077    }
2078 \clist_map_inline:nn { keyans, keyans* } { \__enumext_tmp:n {#1} }
```

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

11.26.1 Store optional arguments of the environments

The idea behind *"storing"* in the ⟨*sequence*⟩ is to have a copy of the structure of the environment in which the key save-ans is being executed so we must capture the optional arguments passed to the levels of the environment in which it is executed and *"storing"* them.

__enumext_store_active_keys:n
__enumext_store_active_keys_vii:n

The functions __enumext_store_active_keys:n and __enumext_store_active_keys_vii:n will be responsible for *"storing"* the ⟨*keys*⟩ filtered from the optional arguments of the environment in which the key save-ans is executed and the levels within this for the enumext and enumext* environments. We will execute this function only if the variable \l__enumext_store_save_key_X_bool is false, that is, the key store-key is not active, establishing the variable \l__enumext_store_save_key_X_tl with the filtered ⟨*keys*⟩.

```
2079 \cs_new_protected:Npn \__enumext_store_active_keys:n #1
2080   {
2081     \bool_if:cF { l__enumext_store_save_key_ \__enumext_level: _bool }
2082       {
2083         \tl_clear:c { l__enumext_save_key_ \__enumext_level: _tl }
2084         \tl_set:ce
2085           { l__enumext_store_save_key_ \__enumext_level: _tl }
2086           { \__enumext_filter_save_key:n {#1} }
2087       }
2088   }
2089 \cs_new_protected:Npn \__enumext_store_active_keys_vii:n #1
2090   {
2091     \bool_if:NF \l__enumext_store_save_key_vii_bool
2092       {
2093         \tl_clear:N \l__enumext_store_save_key_vii_tl
2094         \tl_set:Ne \l__enumext_store_save_key_vii_tl { \__enumext_filter_save_key:n {#1} }
2095       }
2096   }
```

(*End of definition for* __enumext_store_active_keys:n *and* __enumext_store_active_keys_vii:n*.*)

11.26.2 Setting save-key key

Since this list structure will be stored in the ⟨*sequence*⟩ established by the save-ans key when executing \anskey, we will not be able to modify it. The best thing here is to have a key that allows you to modify the optional argument of the list stored in the ⟨*sequence*⟩.

save-key

The values set by this key passed in the optional arguments of the enumext and enumext* environments will override the values of the \l__enumext_store_save_key_X_tl variable set by the functions __enumext_store_active_keys:n and __enumext_store_active_keys_vii:n.

Define the key save-key for all levels of enumext and enumext* environments.

```
2097 \cs_set_protected:Npn \__enumext_tmp:n #1
2098   {
2099     \keys_define:nn { enumext / enumext* }
2100       {
2101         save-key .code:n = \__enumext_parse_save_key_vii:n {##1},
2102         save-key .value_required:n = true,
2103       }
2104     \keys_define:nn { enumext / #1 }
2105       {
2106         save-key .code:n = \__enumext_parse_save_key:n {##1},
2107         save-key .value_required:n = true,
2108       }
2109   }
2110 \clist_map_inline:nn { level-1, level-2, level-3, level-4 } { \__enumext_tmp:n {#1} }
```

(*End of definition for* `save-key`.)

__enumext_parse_save_key:n
__enumext_parse_save_key_vii:n

The functions __enumext_parse_save_key:n and __enumext_parse_save_key_vii:n will be responsible for storing the filtered ⟨*keys*⟩ in the variable \l__enumext_store_save_key_X_tl for enumext and enumext*.

```
2111 \cs_new_protected:Npn \__enumext_parse_save_key:n #1
2112   {
2113     \bool_set_true:c { l__enumext_store_save_key_ \__enumext_level: _bool }
2114     \tl_clear:c { l__enumext_save_key_ \__enumext_level: _tl }
2115     \tl_set:ce
2116       { l__enumext_store_save_key_ \__enumext_level: _tl }
2117       { \__enumext_filter_save_key:n {#1} }
2118   }
2119 \cs_new_protected:Npn \__enumext_parse_save_key_vii:n #1
2120   {
2121     \bool_set_true:N \l__enumext_store_save_key_vii_bool
2122     \tl_clear:N \l__enumext_store_save_key_vii_tl
2123     \tl_set:Ne \l__enumext_store_save_key_vii_tl { \__enumext_filter_save_key:n {#1} }
2124   }
```

(*End of definition for* __enumext_parse_save_key:n *and* __enumext_parse_save_key_vii:n.)

11.26.3 Internal functions to store optional arguments

__enumext_filter_save_key:n
__enumext_filter_save_key_key:n
__enumext_filter_save_key_pair:nn

The function __enumext_filter_save_key:n will be in charge of filtering the ⟨*keys*⟩ we want to *store* in ⟨*sequence*⟩ where {#1} represents the optional value passed to the environment.

```
2125 \cs_new:Npn \__enumext_filter_save_key:n #1
2126   {
2127     \use:e
2128       {
2129         \keyval_parse:NNn
2130           \__enumext_filter_save_key_key:n
2131           \__enumext_filter_save_key_pair:nn {#1}
2132       }
2133   }
```

The function __enumext_filter_save_key_key:n will be responsible for filtering the ⟨*keys*⟩ that are passed *"without value"* by excluding the resume, resume*, no-store and base-fix keys.

```
2134 \cs_new:Npn \__enumext_filter_save_key_key:n #1
2135   {
2136     \str_case:nnF {#1}
2137       {
2138         { resume } {} { resume* } {} { no-store } {} { base-fix } {}
2139       }
2140       { , { \exp_not:n {#1} } }
2141   }
```

The function __enumext_filter_save_key_pair:nn will be responsible for filtering the ⟨*keys*⟩ that are passed *"with value"* by excluding the series, resume, save-ans, save-ref, check-ans, show-ans, save-pos, wrap-ans, mark-ans, wrap-opt, save-sep, mark-ref, mini-env, mini-sep, mini-right and mini-right* keys.

```
2142 \cs_new:Npn \__enumext_filter_save_key_pair:nn #1#2
2143   {
2144     \str_case:nnF {#1}
2145       {
2146         { series   } {} { resume    } {} { save-ans } {} { save-ref   } {}
2147         { save-key } {} { check-ans } {} { show-ans } {} { show-pos   } {}
2148         { wrap-ans } {} { mark-ans  } {} { wrap-opt } {} { save-sep   } {}
2149         { mark-ref } {} { mini-env  } {} { mini-sep } {} { mini-right } {}
2150         { mini-right* } {}
2151       }
2152       { , { \exp_not:n {#1} } = { \exp_not:n {#2} } }
2153   }
```

(*End of definition for* __enumext_filter_save_key:n, __enumext_filter_save_key_key:n, *and* __enumext_filter_-save_key_pair:nn.)

11.26.4 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.

```
2154 \cs_new_protected:Npn \__enumext_store_addto_prop:n #1
2155   {
2156     \prop_gput_if_not_in:cen { g__enumext_ \l__enumext_store_name_tl _prop }
2157       {
2158         \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } + 1 }
2159       }
2160       { #1 }
2161   }
2162 \cs_generate_variant:Nn \__enumext_store_addto_prop:n { V, e }
```

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

11.26.5 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.

```
2163 \cs_new_protected:Npn \__enumext_store_addto_seq:n #1
2164   {
2165     \seq_gput_right:cn { g__enumext_ \l__enumext_store_name_tl _seq } { #1 }
2166   }
2167 \cs_generate_variant:Nn \__enumext_store_addto_seq:n { v, V, e }
```

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

11.26.6 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.

```
2168 \cs_new_protected:Nn \__enumext_store_level_open:
2169   {
2170     \bool_if:NT \l__enumext_check_answers_bool
2171       {
2172         \tl_if_empty:cTF { l__enumext_store_save_key_ \__enumext_level: _tl }
2173           {
2174             \__enumext_store_addto_seq:n
2175               {
2176                 \item \begin{enumext}
2177               }
2178           }
2179           {
2180             \tl_put_left:cn { l__enumext_store_save_key_ \__enumext_level: _tl }
2181               {
2182                 \item \begin{enumext} [
2183               }
2184             \tl_put_right:cn { l__enumext_store_save_key_ \__enumext_level: _tl }
2185               {
2186                 ]
2187               }
2188             \__enumext_store_addto_seq:v { l__enumext_store_save_key_ \__enumext_level: _tl }
2189           }
2190       }
2191   }
2192 \cs_new_protected:Nn \__enumext_store_level_close:
2193   {
2194     \bool_if:NT \l__enumext_check_answers_bool
2195       {
2196         \__enumext_store_addto_seq:n { \end{enumext} }
2197       }
2198   }
```

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

__enumext_store_level_open_vii:
__enumext_store_level_close_vii:

The memorization structure of the list is handled by the functions `\__enumext_store_level_open_vii:` and `\__enumext_store_level_close_vii:` which are executed in the enumext* environment.

```
2199 \cs_new_protected:Nn \__enumext_store_level_open_vii:
2200   {
2201     \bool_if:NT \l__enumext_check_answers_bool
2202       {
2203         \tl_if_empty:NTF \l__enumext_store_save_key_vii_tl
2204           {
2205             \__enumext_store_addto_seq:n
2206               {
2207                 \item \begin{enumext*}
2208               }
2209           }
2210           {
2211             \tl_put_left:Nn \l__enumext_store_save_key_vii_tl
2212               {
2213                 \item \begin{enumext*}[
2214               }
2215             \tl_put_right:Nn \l__enumext_store_save_key_vii_tl
2216               {
2217                 ]
2218               }
2219             \__enumext_store_addto_seq:V \l__enumext_store_save_key_vii_tl
2220           }
2221       }
2222   }
2223 \cs_new_protected:Nn \__enumext_store_level_close_vii:
2224   {
2225     \bool_if:NT \l__enumext_check_answers_bool
2226       {
2227         \__enumext_store_addto_seq:n { \end{enumext*} }
2228       }
2229   }
```

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

11.26.7 Function for show marks and position

__enumext_print_keyans_box:NN
__enumext_print_keyans_box:cc

The function `\__enumext_print_keyans_box:NN` print a box in the left margin with `\l__enumext_-mark_answer_sym_tl` used by the wrap-ans, show-ans and show-pos keys. The function takes two arguments:

```
#1: \l__enumext_labelwidth_X_dim
#2: \l__enumext_labelsep_X_dim
```

```
2230 \cs_new_protected:Nn \__enumext_print_keyans_box:NN
2231   {
2232     \mode_leave_vertical:
2233     \skip_horizontal:n { -\dim_use:N #2 }
2234     \makebox[0pt][ r ]
2235       {
2236         \makebox[ \dim_use:N #1 ][ \l__enumext_mark_position_str ]
2237           {
2238             \tl_use:N \l__enumext_mark_answer_sym_tl
2239           }
2240       }
2241     \skip_horizontal:n { \dim_use:N #2 }
2242   }
2243 \cs_generate_variant:Nn \__enumext_print_keyans_box:NN { cc }
```

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

11.27 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 similar behaviour to that of \item in the enumext and enumext* environments executed as follows: \anskey[⟨*key = val*⟩]{⟨*content*⟩} so first we'll add the keys break-col, item-join, item-star, item-sym* and item-pos*.

```
2244 \keys_define:nn { enumext / anskey }
2245   {
2246     break-col .bool_set:N = \l__enumext_store_columns_break_bool,
2247     break-col .default:n  = true,
```

```
2248      break-col .value_forbidden:n = true,
2249      item-join .int_set:N  = \l__enumext_store_item_join_int,
2250      item-join .value_required:n = true,
2251      item-star .bool_set:N = \l__enumext_store_item_star_bool,
2252      item-star .default:n  = true,
2253      item-star .value_forbidden:n = true,
2254      item-sym* .tl_set:N   = \l__enumext_store_item_symbol_tl,
2255      item-sym* .value_required:n = true,
2256      item-pos* .dim_set:N  = \l__enumext_store_item_symbol_sep_dim,
2257      item-pos* .value_required:n = true,
2258      unknown   .code:n     = { \__enumext_anskey_unknown:n {#1} },
2259    }
```

The ⟨keys⟩ are stored in \l_keys_key_str and the value (if any) is passed as an argument to the function __enumext_anskey_unknown:n.

```
2260 \cs_new_protected:Npn \__enumext_anskey_unknown:n #1
2261   {
2262     \exp_args:NV \__enumext_anskey_unknown:nn \l_keys_key_str {#1}
2263   }
2264 \cs_new_protected:Npn \__enumext_anskey_unknown:nn #1 #2
2265   {
2266     \tl_if_blank:nTF {#2}
2267       {
2268         \msg_error:nnn { enumext } { anskey-cmd-key-unknown } {#1}
2269       }
2270       {
2271         \msg_error:nnnn { enumext } { anskey-cmd-key-value-unknown } {#1} {#2}
2272       }
2273   }
```

💣 The \anskey command will only be present when using the save-ans key in enumext and enumext* environments, otherwise it will return an error.

\anskey We will first call the function __enumext_anskey_safe_outer: to be sure where we execute the command, then we will check the state of the variable \l__enumext_check_answers_bool set by the key no-store, if is true we will increment \g__enumext_item_anskey_int for the internal *"check answer"* system and execute the function __enumext_anskey_safe_inner:n to ensure that the command is not nested and that the argument is not empty, finally search the [⟨key = val⟩] and call the function __enumext_store_anskey_code:n.

```
2274 \NewDocumentCommand \anskey { o +m }
2275   {
2276     \__enumext_anskey_safe_outer:
2277     \group_begin:
2278       \bool_if:NT \l__enumext_check_answers_bool
2279         {
2280           \tl_if_novalue:nF {#1}
2281             {
2282               \keys_set:nn { enumext / anskey } {#1}
2283             }
2284           \tl_if_blank:nTF {#2}
2285             {
2286               \msg_error:nn { enumext } { anskey-empty-arg }
2287             }
2288             {
2289               \__enumext_anskey_safe_inner:
2290               \__enumext_store_anskey_code:n {#2}
2291             }
2292         }
2293     \group_end:
2294   }
```

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

11.27.1 Internal functions for the command

__enumext_anskey_safe_outer:
__enumext_anskey_safe_inner:

The __enumext_store_anskey_safe_outer: function will return the appropriate messages when the command is executed outside the environment in which the save-ans key was activated.

```
2295 \cs_new_protected:Nn \__enumext_anskey_safe_outer:
2296   {
2297     \bool_if:NF \l__enumext_store_active_bool
2298       {
2299         \msg_error:nnnn { enumext } { anskey-wrong-place }{ anskey }{ enumext }
```

```
2300          }
2301        \int_compare:nNnT { \l__enumext_keyans_level_int } = { 1 }
2302          {
2303            \msg_error:nnnn { enumext } { command-wrong-place }{ anskey }{ keyans }
2304          }
2305        \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
2306          {
2307            \msg_error:nnnn { enumext } { command-wrong-place }{ anskey }{ keyans* }
2308          }
2309        \int_compare:nNnT { \l__enumext_keyans_pic_level_int } = { 1 }
2310          {
2311            \msg_error:nnnn { enumext } { command-wrong-place }{ anskey }{ keyanspic }
2312          }
2313      }
```

The `\__enumext_anskey_safe_inner:` function will first check if the command is nested, if preceded by a not numbered `\item` or if it is in *math mode* returning the appropriate messages.

```
2314  \cs_new_protected:Nn \__enumext_anskey_safe_inner:
2315    {
2316      \int_incr:N \l__enumext_anskey_level_int
2317      \int_compare:nNnT { \l__enumext_anskey_level_int } > { 1 }
2318        {
2319          \msg_error:nn { enumext } { anskey-nested }
2320        }
2321      \bool_if:NF \l__enumext_item_number_bool
2322        {
2323          \msg_error:nn { enumext } { anskey-unnumber-item }
2324        }
2325      \mode_if_math:T
2326        {
2327          \msg_error:nne { enumext } { anskey-math-mode } { \c_backslash_str anskey }
2328        }
2329    }
```

(*End of definition for* `\__enumext_anskey_safe_outer:` *and* `\__enumext_anskey_safe_inner:`.)

`\__enumext_store_anskey_code:n`   The internal function `\__enumext_store_anskey_code:n` first we pass the ⟨*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*⟩.

```
2330  \cs_new_protected:Npn \__enumext_store_anskey_code:n #1
2331    {
2332      \int_gincr:N \g__enumext_item_anskey_int
2333      \__enumext_store_addto_prop:n {#1}
2334      \bool_if:NT \l__enumext_store_ref_key_bool
2335        {
2336          \__enumext_store_internal_ref:
2337        }
2338      \__enumext_anskey_show_wrap_left:n { #1 }
```

Now we start processing the [⟨*key = val*⟩] 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 the ⟨*keys*⟩, if the `break-col` key is present and the command is running under `enumext` (not in `enumext*`) we will add `\columnbreak` and then `\item`.

```
2339      \tl_clear:N \l__enumext_store_anskey_arg_tl
2340      \bool_lazy_and:nnT
2341        { \bool_if_p:N \l__enumext_store_columns_break_bool }
2342        { \bool_not_p:n { \l__enumext_starred_bool } }
2343        {
2344          \tl_put_left:Nn \l__enumext_store_anskey_arg_tl { \columnbreak }
2345        }
2346      \tl_put_right:Nn \l__enumext_store_anskey_arg_tl { \item }
```

If the `item-join` key is present and the command is running under `enumext*` we will add (⟨*number*⟩) to `\l__enumext_store_anskey_arg_tl`.

```
2347      \bool_lazy_and:nnT
2348        { \bool_not_p:n { \l__enumext_starred_bool } }
2349        { \int_compare_p:nNn { \l__enumext_store_item_join_int } > { 1 } }
2350        {
2351          \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2352            {
2353              ( \exp_not:V \l__enumext_store_item_join_int )
```

```
2354                    }
2355             }
```

And now we will review the keys `item-star`, `item-sym*` and `item-pos*` and pass them to `\l__enumext_store_anskey_arg_tl` along with the ⟨*argument*⟩ for `\anskey` or ⟨*body*⟩ for `anskey*`.

```
2356        \bool_if:NTF \l__enumext_store_item_star_bool
2357          {
2358            \tl_put_right:Nn \l__enumext_store_anskey_arg_tl { * }
2359            \tl_if_empty:NF \l__enumext_store_item_symbol_tl
2360              {
2361                \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2362                  {
2363                    [ \exp_not:V \l__enumext_store_item_symbol_tl ]
2364                  }
2365              }
2366            \dim_compare:nT
2367              {
2368                \l__enumext_store_item_symbol_sep_dim != \c_zero_dim
2369              }
2370              {
2371                \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2372                  {
2373                    [ \exp_not:V \l__enumext_store_item_symbol_sep_dim ]
2374                  }
2375              }
2376            \tl_put_right:Nn \l__enumext_store_anskey_arg_tl {#1}
2377          }
2378          {
2379            \tl_put_right:Nn \l__enumext_store_anskey_arg_tl {#1}
2380          }
```

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

```
2381        \bool_lazy_and:nnT
2382          { \bool_if_p:N \l__enumext_store_ref_key_bool }
2383          { \bool_if_p:N \l__enumext_hyperref_bool }
2384          {
2385            \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2386              {
2387                \hfill \exp_not:N \hyperlink { \exp_not:V \l__enumext_newlabel_arg_one_tl }
2388                    { \exp_not:V \l__enumext_mark_ref_sym_tl }
2389              }
2390          }
2391        \__enumext_store_addto_seq:V \l__enumext_store_anskey_arg_tl
2392      }
```

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

`\__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`.

```
2393 \cs_new_protected:Nn \__enumext_store_internal_ref:
2394   {
2395     \cs_set_protected:Npn \__enumext_tmp:n ##1
2396       {
2397         \tl_set_eq:cc { l__enumext_label_copy_##1_tl } { l__enumext_label_##1_tl }
2398         \tl_reverse:c { l__enumext_label_copy_##1_tl }
2399         \tl_remove_once:cn { l__enumext_label_copy_##1_tl } { . }
2400         \tl_reverse:c { l__enumext_label_copy_##1_tl }
2401       }
2402     \clist_map_inline:nn { i, ii, iii, iv, vii } { \__enumext_tmp:n {##1} }
2403     \cs_set:Npn \__enumext_tmp:n ##1
2404       { . \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` or `anskey*` is running alone in it or if it is running in a nested `enumext` environment within the starting environment.

```
2405        \bool_lazy_all:nT
2406          {
2407            { \bool_if_p:N \g__enumext_starred_bool }
2408            { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
```

```
2409            }
2410            {
2411                \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2412                    { \tl_use:N \l__enumext_label_copy_vii_tl }
2413            }
2414        \bool_lazy_all:nT
2415            {
2416                { \bool_if_p:N \l__enumext_standar_bool }
2417                { \bool_if_p:N \g__enumext_starred_bool }
2418                { \int_compare_p:nNn { \l__enumext_level_int } > { 0 } }
2419            }
2420            {
2421                \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2422                    {
2423                        \tl_use:N \l__enumext_label_copy_vii_tl
2424                        \int_step_function:nnN { 1 } { \l__enumext_level_int } \__enumext_tmp:n
2425                    }
2426            }
```

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

```
2427        \bool_lazy_all:nT
2428            {
2429                { \bool_if_p:N \l__enumext_standar_bool }
2430                { \int_compare_p:nNn { \l__enumext_level_int } > { 0 } }
2431                { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
2432                { \bool_not_p:n { \l__enumext_starred_bool } }
2433            }
2434            {
2435                \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2436                    {
2437                        \tl_use:N \l__enumext_label_copy_i_tl
2438                        \int_step_function:nnN { 2 } { \l__enumext_level_int } \__enumext_tmp:n
2439                    }
2440            }
2441        \cs_set:Npn \__enumext_tmp:n ##1
2442            { \tl_use:c { l__enumext_label_copy_ \int_to_roman:n {##1} _tl } }
2443        \bool_lazy_all:nT
2444            {
2445                { \bool_if_p:N \l__enumext_standar_bool }
2446                { \int_compare_p:nNn { \l__enumext_level_int } > { 0 } }
2447                { \bool_not_p:n { \g__enumext_starred_bool } }
2448                { \int_compare_p:nNn { \l__enumext_level_h_int } > { 0 } }
2449            }
2450            {
2451                \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2452                    {
2453                        \int_step_function:nnN { 1 } { \l__enumext_level_int } \__enumext_tmp:n
2454                        . \tl_use:N \l__enumext_label_copy_vii_tl
2455                    }
2456            }
```

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

```
2457        \tl_put_right:Ne \l__enumext_newlabel_arg_one_tl
2458            {
2459                \l__enumext_store_name_tl \c_colon_str
2460                \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } }
2461            }
```

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

```
2462        \tl_put_right:Ne \l__enumext_write_aux_file_tl
2463            {
2464                \__enumext_newlabel:nn
2465                    { \exp_not:V \l__enumext_newlabel_arg_one_tl }
2466                    { \l__enumext_newlabel_arg_two_tl }
2467            }
2468        \l__enumext_write_aux_file_tl
2469    }
```

(*End of definition for* __enumext_store_internal_ref:.)

__enumext_anskey_show_wrap_arg:n

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

```
2470 \cs_new_protected:Npn \__enumext_anskey_show_wrap_arg:n #1
2471   {
2472     \par
2473     \bool_if:NT \l__enumext_starred_bool
2474       {
2475         \cs_set:Nn \__enumext_level: { vii }
2476       }
2477     \__enumext_print_keyans_box:cc
2478       { l__enumext_labelwidth_ \__enumext_level: _dim }
2479       { l__enumext_labelsep_ \__enumext_level: _dim }
2480     \__enumext_anskey_wrapper:n { #1 }
2481   }
```

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

__enumext_anskey_show_wrap_left:n

The function __enumext_anskey_show_wrap_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 and the ⟨*body*⟩ in anskey* on the right side when using the show-ans key.

```
2482 \cs_new_protected:Npn \__enumext_anskey_show_wrap_left:n #1
2483   {
2484     \bool_if:NT \l__enumext_show_answer_bool
2485       {
2486         \__enumext_anskey_show_wrap_arg:n { #1 }
2487       }
2488     \bool_if:NT \l__enumext_show_position_bool
2489       {
2490         \tl_set:Ne \l__enumext_mark_answer_sym_tl
2491           {
2492             \group_begin:
2493             \exp_not:N \normalfont
2494             \exp_not:N \footnotesize [ \int_eval:n
2495               {
2496                 \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
2497               }
2498             ]
2499             \group_end:
2500           }
2501         \__enumext_anskey_show_wrap_arg:n { #1 }
2502       }
2503   }
```

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

11.28   The environment anskey*

Managing *verbatim content* in an environment is quite complicated, I learned that when creating the scontents package, so to be able to have support at this point it is best to play a little with the internal code of scontents and *hooks*. Some considerations I should have here before implementing this:

– If some package, class or user has defined the environment with the same name somewhere in the document it would be a problem, you would not know what argument has been passed to store-env, if you are using the key print-env or the write-out key, sure, I can detect and modify it within the enumext and enumext* environments, but it would look strange not to have some keys available when running within these environments.

– A better (perhaps a bit paranoid) option is to define it within the environment in which the save-ans key is executed. and have it available only when that key is executed, here I would have absolute control of the ⟨*keys*⟩ and I make sure that write-out is not used, then using *hooks after* I undefine it and using *hook before* I check if it has been created by any package, class or user and I return a error, then the user will have to see how to solve the problem.

__enumext_undefine_anskey_env:

The function __enumext_undefine_anskey_env: will undefine the environment anskey* and will be passed to the function __enumext_execute_after_env: (§11.25) which is executed after the environment in which the key save-ans is active.

```
2504 \cs_new_protected:Nn \__enumext_undefine_anskey_env:
2505   {
2506     \cs_undefine:c { anskey* }
2507     \cs_undefine:c { endanskey* }
2508     \cs_undefine:c { __scontents_anskey*_env_begin: }
```

```
2509        \cs_undefine:c { __scontents_anskey*_env_end: }
2510      }
```

Detection of the anskey* environment outside the enumext and enumext* environments.

```
2511  \__enumext_before_env:nn { enumext }
2512    {
2513      \bool_lazy_and:nnT
2514        { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
2515        { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
2516        {
2517          \cs_if_free:cF { __scontents_anskey*_env_begin: }
2518            {
2519              \msg_error:nnn { enumext } { anskey-env-error } { anskey* }
2520            }
2521        }
2522    }
2523  \__enumext_before_env:nn { enumext* }
2524    {
2525      \bool_lazy_and:nnT
2526        { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
2527        { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
2528        {
2529          \cs_if_free:cF { __scontents_anskey*_env_begin: }
2530            {
2531              \msg_error:nnn { enumext } { anskey-env-error } { anskey* }
2532            }
2533        }
2534    }
```

Detection of the anskey* environment inside the keyans, keyans* and keyanspic environments, if preceded by a not numbered \item or if it is in *math mode* returning the appropriate messages.

```
2535  \__enumext_before_env:nn { anskey* }
2536    {
2537      \int_compare:nNnT { \l__enumext_keyans_level_int } = { 1 }
2538        {
2539          \msg_error:nnn { enumext } { anskey-env-wrong }{ keyans }
2540        }
2541      \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
2542        {
2543          \msg_error:nnn { enumext } { anskey-env-wrong } { keyans* }
2544        }
2545      \int_compare:nNnT { \l__enumext_keyans_pic_level_int } = { 1 }
2546        {
2547          \msg_error:nnn { enumext } { anskey-env-wrong } { keyanspic }
2548        }
2549      \bool_if:NF \l__enumext_item_number_bool
2550        {
2551          \msg_error:nn { enumext } { anskey-unnumber-item }
2552        }
2553      \mode_if_math:T
2554        {
2555          \msg_error:nnn { enumext } { anskey-math-mode } { anskey* }
2556        }
2557    }
```

(*End of definition for* __enumext_undefine_anskey_env:.)

anskey*
__enumext_anskey_env_make:n
__enumext_anskey_env_make:V
__enumext_anskey_env_define_keys:
__enumext_rescan_anskey_env:n

The function __enumext_anskey_env_make:n creates the environment anskey* (*custom version* of scontents environment) by setting the initial keys store-env={⟨*store name*⟩} and print-env=false. To maintain the *scope* of the environment and that it is only active when the key save-ans is active we will pass this function to the function __enumext_storing_exec: (§11.24.1) and we will execute it only if the variable \l__enumext_anskey_env_bool is true, with this we prevent it from being executed again when the environment is nested and the key save-ans is active, which returns an error for part of the package scontents.

```
2558  \cs_new_protected:Npn \__enumext_anskey_env_make:n #1
2559    {
2560      \bool_if:NT \l__enumext_anskey_env_bool
2561        {
2562          \newenvsc{anskey*}[store-env=#1,print-env=false]
2563          \__enumext_anskey_env_exec:
2564        }
```

```
2565    }
2566 \cs_generate_variant:Nn \__enumext_anskey_env_make:n { V }
```

The function __enumext_anskey_env_define_keys: will add the keys break-col, item-join, item-join, item-star, item-sym* and item-pos* and will leave the keys print-env, store-env and write-out undefined. We will apply this function using the *hook* function __enumext_before_-env:nn.

```
2567 \cs_new_protected:Nn \__enumext_anskey_env_define_keys:
2568    {
2569      \keys_define:nn { scontents / scontents }
2570        {
2571          break-col .bool_gset:N = \g__enumext_store_columns_break_bool,
2572          break-col .default:n   = true,
2573          break-col .value_forbidden:n = true,
2574          item-join .int_gset:N  = \g__enumext_store_item_join_int,
2575          item-join .value_required:n = true,
2576          item-star .bool_gset:N = \g__enumext_store_item_star_bool,
2577          item-star .default:n   = true,
2578          item-star .value_forbidden:n = true,
2579          item-sym* .tl_gset:N   = \g__enumext_store_item_symbol_tl,
2580          item-sym* .value_required:n = true,
2581          item-pos* .dim_gset:N  = \g__enumext_store_item_symbol_sep_dim,
2582          item-pos* .value_required:n = true,
2583          print-env .undefine:,
2584          store-env .undefine:,
2585          write-out .undefine:,
2586          unknown   .code:n       = { \__enumext_anskey_env_unknown:n {##1} },
2587        }
2588    }
```

The ⟨keys⟩ are stored in \l_keys_key_str and the value (if any) is passed as an argument to the function __enumext_anskey_env_unknown:n.

```
2589 \cs_new_protected:Npn \__enumext_anskey_env_unknown:n #1
2590    {
2591      \exp_args:NV \__enumext_anskey_env_unknown:nn \l_keys_key_str {#1}
2592    }
2593 \cs_new_protected:Npn \__enumext_anskey_env_unknown:nn #1#2
2594    {
2595      \tl_if_blank:nTF {#2}
2596        {
2597          \msg_error:nnn { enumext } { anskey-env-key-unknown } {#1}
2598        }
2599        {
2600          \msg_error:nnnn { enumext } { anskey-env-key-value-unknown } {#1} {#2}
2601        }
2602    }
```

The function __enumext_anskey_env_reset_keys: will leave the keys break-col, item-join, item-join, item-star, item-sym* and item-pos* undefined. We will apply this function using the *hook* function __enumext_after_env:nn.

```
2603 \cs_new_protected:Nn \__enumext_anskey_env_reset_keys:
2604    {
2605      \keys_define:nn { scontents / scontents }
2606        {
2607          break-col .undefine:,
2608          item-join .undefine:,
2609          item-star .undefine:,
2610          item-sym* .undefine:,
2611          item-pos* .undefine:,
2612          write-out .code:n     = {
2613                                   \bool_set_false:N \l__scontents_storing_bool
2614                                   \bool_set_true:N  \l__scontents_writing_bool
2615                                   \tl_set:Nn \l__scontents_fname_out_tl {##1}
2616                                 },
2617          write-out .value_required:n = true,
2618          print-env .meta:nn    = { scontents } { print-env = ##1 },
2619          print-env .default:n = true,
2620          store-env .meta:nn   = { scontents } { store-env = ##1 },
2621          unknown   .code:n     = { \__scontents_parse_environment_keys:n {##1} },
2622        }
2623    }
```

The function `\__enumext_rescan_anskey_env:n` will be responsible for bringing the ⟨*body*⟩ of the environment saved in the sequence `\g__scontents_name_`⟨*store name*⟩`_seq` to pass it to our *sequence* and *prop list*.

```
2624 \cs_new_protected:Npn \__enumext_rescan_anskey_env:n #1
2625   {
2626     \group_begin:
2627       \int_set:Nn \tex_newlinechar:D { `\^^J }
2628       \__scontents_rescan_tokens:x
2629         {
2630           \endgroup % This assumes \catcode`\\=0... Things might go off otherwise.
2631           #1
2632         }
2633   }
```

(*End of definition for* `anskey*` *and others. This function is documented on page 13.*)

`\__enumext_anskey_env_exec:`   The function `\__enumext_anskey_env_exec:` will be responsible for processing all the code necessary for the execution of the environment. The first thing will be to add our ⟨*keys*⟩.

```
2634 \cs_new_protected:Nn \__enumext_anskey_env_exec:
2635   {
2636     \__enumext_before_env:nn { anskey* }
2637       {
2638         \__enumext_anskey_env_define_keys:
2639       }
```

Now we will execute our actions after the `anskey*` environment is closed. We'll fetch the contents of the *environment body* that is now saved in `\g__scontents_name_`⟨*store name*⟩`_seq` and store it in the variable `\l__enumext_store_anskey_env_tl` then we execute the rest of the functions.

```
2640     \hook_if_empty:nF {env/anskey*/after}
2641       {
2642         \hook_gremove_code:nn {env/anskey*/after} { * }
2643       }
2644     \__enumext_after_env:nn { anskey* }
2645       {
2646         \__enumext_anskey_env_save_keys:
2647         \tl_clear:N \l__enumext_store_anskey_env_tl
2648         \tl_clear:N \l__enumext_store_anskey_opt_tl
2649         \bool_if:NT \l__enumext_check_answers_bool
2650           {
2651             \tl_gset:Ne \l__enumext_store_anskey_env_tl
2652               {
2653                 \seq_item:ce { g__scontents_name_ \l__enumext_store_name_tl _seq } { -1 }
2654               }
2655             \regex_match:nVTF
2656               { ^\s* \z | ^\s* \u{c__scontents_hidden_space_str} \z }
2657               \l__enumext_store_anskey_env_tl
2658               {
2659                 \msg_error:nn { enumext } { anskey-empty-arg }
2660               }
2661               {
2662                 \__enumext_anskey_env_store:
2663               }
2664           }
2665         \__enumext_anskey_env_clean_vars:
2666         \__enumext_anskey_env_reset_keys:
2667       }
2668   }
```

💡 The use of `\hook_gremove_code:nn` is necessary here, otherwise the {⟨*code*⟩} passed to `\__enumext_after_-env:nn`{`anskey*`} will be accumulated for each execution. The last function `\__enumext_anskey_env_reset_-keys:` is necessary so as not to hinder any `scontents` environment running within `enumext` or `enumext*`.

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

`\__enumext_anskey_env_save_keys:`
`\__enumext_anskey_env_store:`
`\__enumext_anskey_env_clean_vars:`

The function `\__enumext_anskey__env_keys:` processing the [⟨*key = val*⟩] passed to the environment and save this in the variable `\l__enumext_store_anskey_opt_tl`. If the `break-col` key is present and the environment is running under `enumext` (not in `enumext*`) we will add the key `break-col`.

```
2669 \cs_new_protected:Nn \__enumext_anskey_env_save_keys:
2670   {
2671     \bool_lazy_and:nnT
2672       { \bool_if_p:N \g__enumext_store_columns_break_bool }
```

```
2673          { \bool_not_p:n { \l__enumext_starred_bool } }
2674          {
2675              \tl_put_left:Ne \l__enumext_store_anskey_opt_tl { ,break-col, }
2676          }
```

If the `item-join` key is present and the command is running under `enumext*` we will add to `\l__enumext_store_anskey_opt_tl`.

```
2677          \bool_lazy_and:nnT
2678          { \bool_not_p:n { \l__enumext_starred_bool } }
2679          { \int_compare_p:nNn { \g__enumext_store_item_join_int } > { 1 } }
2680          {
2681              \tl_put_left::Ne \l__enumext_store_anskey_opt_tl
2682              {
2683                  ,item-join = \exp_not:V \g__enumext_store_item_join_int,
2684              }
2685          }
```

And now we will review the keys `item-star`, `item-sym*` and `item-pos*` and pass them to `\l__enumext_store_anskey_opt_tl`.

```
2686          \bool_if:NT \g__enumext_store_item_star_bool
2687          {
2688              \tl_put_left:Ne \l__enumext_store_anskey_opt_tl
2689              {
2690                  ,item-star,
2691              }
2692              \tl_if_empty:NF \g__enumext_store_item_symbol_tl
2693              {
2694                  \tl_put_left:Ne \l__enumext_store_anskey_opt_tl
2695                  {
2696                      ,item-sym* = \exp_not:V \g__enumext_store_item_symbol_tl,
2697                  }
2698              }
2699              \dim_compare:nT
2700              {
2701                  \g__enumext_store_item_symbol_sep_dim != \c_zero_dim
2702              }
2703              {
2704                  \tl_put_left:Ne \l__enumext_store_anskey_opt_tl
2705                  {
2706                      ,item-pos* = \exp_not:V \g__enumext_store_item_symbol_sep_dim,
2707                  }
2708              }
2709          }
2710      }
```

The function `\__enumext_anskey_env_store:` will be responsible for storing the content of the environment using the functions `\__enumext_store_anskey_code:n` and `\__enumext_rescan_anskey_env:n`.

```
2711 \cs_new_protected:Nn \__enumext_anskey_env_store:
2712      {
2713          \group_begin:
2714          \tl_if_empty:NTF \l__enumext_store_anskey_opt_tl
2715          {
2716              \exp_args:Ne
2717                  \__enumext_store_anskey_code:n
2718                  {
2719                      \__enumext_rescan_anskey_env:n { \l__enumext_store_anskey_env_tl }
2720                  }
2721          }
2722          {
2723              \keys_set_known:nV { enumext / anskey } \l__enumext_store_anskey_opt_tl
2724              \exp_args:Ne
2725                  \__enumext_store_anskey_code:n
2726                  {
2727                      \__enumext_rescan_anskey_env:n { \l__enumext_store_anskey_env_tl }
2728                  }
2729          }
2730          \group_end:
2731      }
```

The function `\__enumext_anskey_env_clean_vars:` will return the global variables used by the ⟨keys⟩ to their initial state.

```
2732 \cs_new_protected:Nn \__enumext_anskey_env_clean_vars:
2733   {
2734     \bool_gset_false:N \g__enumext_store_columns_break_bool
2735     \int_gzero:N       \g__enumext_store_item_join_int
2736     \bool_gset_false:N \g__enumext_store_item_star_bool
2737     \tl_gclear:N       \g__enumext_store_item_symbol_tl
2738     \dim_gzero:N       \g__enumext_store_item_symbol_sep_dim
2739   }
```

(*End of definition for* __enumext_anskey_env_save_keys: , __enumext_anskey_env_store: , *and* __enumext_anskey_-env_clean_vars:.)

11.29 Common functions for keyans, keyans* and keyanspic

11.29.1 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_current_label_tl variable, which will be passed to the ⟨*prop list*⟩ defined by the save-ans key using the __enumext_store_addto_-prop:V.

```
2740 \cs_new_protected:Npn \__enumext_keyans_addto_prop:n #1
2741   {
2742     \tl_clear:N \l__enumext_store_current_label_tl
2743     \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
2744       {
2745         \tl_put_right:Ne \l__enumext_store_current_label_tl { \l__enumext_label_vi_tl }
2746       }
2747       {
2748         \tl_put_right:Ne \l__enumext_store_current_label_tl { \l__enumext_label_v_tl }
2749       }
2750     \tl_if_novalue:nF { #1 }
2751       {
2752         % Set save-sep
2753         \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_tl
2754           {
2755             \tl_put_right:Ne \l__enumext_store_current_label_tl { \l__enumext_store_keyans_item_op
2756           }
2757         \tl_put_right:Ne \l__enumext_store_current_label_tl { #1 }
2758       }
2759     \__enumext_store_addto_prop:V \l__enumext_store_current_label_tl
2760   }
```

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

11.29.2 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.

```
2761 \cs_new_protected:Nn \__enumext_keyans_store_ref:
2762   {
2763     \bool_if:NT \l__enumext_store_ref_key_bool
2764       {
2765         \cs_set_protected:Npn \__enumext_tmp:n ##1
2766           {
2767             \tl_set_eq:cc { l__enumext_label_copy_##1_tl } { l__enumext_label_##1_tl }
2768             \tl_reverse:c { l__enumext_label_copy_##1_tl }
2769             \tl_remove_once:cn { l__enumext_label_copy_##1_tl } { . }
2770             \tl_reverse:c { l__enumext_label_copy_##1_tl }
2771           }
2772         \clist_map_inline:nn { i, v, vi, vii, viii } { \__enumext_tmp:n {##1} }
2773         \__enumext_keyans_store_ref_aux_i:
2774       }
2775   }
```

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`.

```
2776  \cs_new_protected:Nn \__enumext_keyans_store_ref_aux_i:
2777    {
2778      \bool_if:NT \g__enumext_starred_bool
2779        {
2780          \tl_set_eq:NN \l__enumext_label_copy_i_tl \l__enumext_label_copy_vii_tl
2781        }
2782      \int_compare:nNnT { \l__enumext_keyans_pic_level_int } = { 1 }
2783        {
2784          \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2785            { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_vi_tl }
2786        }
2787      \int_compare:nNnT { \l__enumext_keyans_level_int } = { 1 }
2788        {
2789          \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2790            { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_v_tl }
2791        }
2792      \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
2793        {
2794          \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2795            { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_viii_tl }
2796        }
2797      \tl_put_right:Ne \l__enumext_newlabel_arg_one_tl
2798        {
2799          \l__enumext_store_name_tl \c_colon_str
2800          \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } }
2801        }
2802      \__enumext_keyans_store_ref_aux_ii:
2803    }
```

Now auxiliary function `\__enumext_keyans_store_ref_aux_ii:` save the result in the variable `\l__enumext_write_aux_file_tl` and finally we write in the `.aux` file.

```
2804  \cs_new_protected:Nn \__enumext_keyans_store_ref_aux_ii:
2805    {
2806      \tl_put_right:Ne \l__enumext_write_aux_file_tl
2807        {
2808          \__enumext_newlabel:nn
2809            { \exp_not:V \l__enumext_newlabel_arg_one_tl }
2810            { \l__enumext_newlabel_arg_two_tl }
2811        }
2812      \l__enumext_write_aux_file_tl
2813    }
```

(*End of definition for* `\__enumext_keyans_store_ref:` , `\__enumext_keyans_store_ref_aux_i:` , *and* `\__enumext_keyans_store_ref_aux_ii:` .)

11.29.3 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_current_label_tl` variable to the sequence defined by the `save-ans` key.

```
2814  \cs_new_protected:Npn \__enumext_keyans_addto_seq:n #1
2815    {
2816      \tl_clear:N \l__enumext_store_current_label_tl
2817      \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
2818        {
2819          \tl_put_right:Ne \l__enumext_store_current_label_tl { \item \l__enumext_label_vi_tl }
2820        }
2821        {
2822          \tl_put_right:Ne \l__enumext_store_current_label_tl { \item \l__enumext_label_v_tl }
2823        }
2824      \tl_if_novalue:nF { #1 }
2825        {
2826          \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_tl
2827            {
2828              \tl_put_right:Ne \l__enumext_store_current_label_tl
2829                {
2830                  \l__enumext_store_keyans_item_opt_sep_tl
```

```
2831                        }
2832                    }
2833                \tl_put_right:Ne \l__enumext_store_current_label_tl { #1 }
2834            }
2835        \__enumext_keyans_addto_seq_link:
2836    }
```

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_current_label_tl into the global variable \g__enumext_check_ans_item_tl to be used by the function __enumext_check_starred_cmd:n and increment the value of the integer variable \g__enumext_item_anskey_int handled by the check-ans key.

```
2837 \cs_new_protected:Nn \__enumext_keyans_addto_seq_link:
2838    {
2839        \bool_lazy_and:nnT
2840            { \bool_if_p:N \l__enumext_store_ref_key_bool }
2841            { \bool_if_p:N \l__enumext_hyperref_bool }
2842            {
2843                \tl_put_right:Ne \l__enumext_store_current_label_tl
2844                    {
2845                        \hfill \exp_not:N \hyperlink
2846                            {
2847                                \exp_not:V \l__enumext_newlabel_arg_one_tl
2848                            }
2849                            { \exp_not:V \l__enumext_mark_ref_sym_tl }
2850                    }
2851            }
2852        \__enumext_store_addto_seq:V \l__enumext_store_current_label_tl
2853        \bool_if:NT \l__enumext_check_answers_bool
2854            {
2855                \int_gincr:N \g__enumext_item_anskey_int
2856            }
2857    }
```

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

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

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

__enumext_keyans_show_left:n
__enumext_keyans_show_ans:
__enumext_keyans_show_pos:
__enumext_keyans_show_item_opt:

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

```
2858 \cs_new_protected:Npn \__enumext_keyans_show_left:n #1
2859    {
2860        \tl_if_novalue:nF { #1 }
2861            {
2862                \tl_set:Ne \l__enumext_store_current_opt_arg_tl { #1 }
2863            }
2864        \bool_if:NT \l__enumext_show_answer_bool
2865            {
2866                \__enumext_keyans_show_ans:
2867            }
2868        \bool_if:NT \l__enumext_show_position_bool
2869            {
2870                \__enumext_keyans_show_pos:
2871            }
2872    }
2873 \cs_new_protected:Nn \__enumext_keyans_show_item_opt:
2874    {
2875        \tl_if_empty:NF \l__enumext_store_current_opt_arg_tl
2876            {
2877                \bool_lazy_or:nnT
2878                    { \bool_if_p:N \l__enumext_show_answer_bool }
2879                    { \bool_if_p:N \l__enumext_show_position_bool }
2880                    {
2881                        \__enumext_keyans_wrapper_opt:n { \l__enumext_store_current_opt_arg_tl } \c_space_tl
2882                    }
2883            }
2884    }
2885 \cs_new_protected:Nn \__enumext_keyans_show_ans:
```

```
2886      {
2887        \tl_put_left:Nn \l__enumext_label_v_tl
2888          {
2889            \__enumext_print_keyans_box:NN
2890              \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
2891          }
2892      }
2893  \cs_new_protected:Nn \__enumext_keyans_show_pos:
2894      {
2895        \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
2896          {
2897            \tl_set:Ne \l__enumext_mark_answer_sym_tl
2898              {
2899                \group_begin:
2900                \exp_not:N \normalfont
2901                \exp_not:N \footnotesize [ \int_eval:n
2902                  {
2903                    \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
2904                  }
2905                  ]
2906                \group_end:
2907              }
2908          }
2909          {
2910            \tl_set:Ne \l__enumext_mark_answer_sym_tl
2911              {
2912                \group_begin:
2913                \exp_not:N \normalfont
2914                \exp_not:N \footnotesize [ \int_eval:n
2915                  {
2916                    \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } + 1
2917                  }
2918                  ]
2919                \group_end:
2920              }
2921          }
2922        \tl_put_left:Nn \l__enumext_label_v_tl
2923          {
2924            \__enumext_print_keyans_box:NN
2925              \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
2926          }
2927      }
```

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

11.30   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*
```
2928  \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
2929      {
2930        \keys_define:nn { enumext / #1 }
2931          {
2932            item-sym* .tl_set:c  = { l__enumext_item_symbol_#2_tl },
2933            item-sym* .value_required:n = true,
2934            item-sym* .initial:n  = {$\star$},
2935            item-pos* .dim_set:c  = { l__enumext_item_symbol_sep_#2_dim },
2936            item-pos* .value_required:n = true,
2937          }
2938      }
2939  \clist_map_inline:nn
2940      {
2941        {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {enumext*}{vii}
2942      }
2943    { \__enumext_tmp:nn #1 }
```

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

11.31   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.

```
2944  \cs_new_protected:Nn \__enumext_footnotetext:nn
2945    {
2946      \footnotetext[#1]{#2}
2947    }
2948  \cs_new_protected:Nn \__enumext_renew_footnote:
2949    {
2950      \seq_gclear:N \g__enumext_footnote_arg_seq
2951      \seq_gclear:N \g__enumext_footnote_int_seq
2952      \RenewDocumentCommand \footnote { o +m }
2953        {
2954          \tl_if_novalue:nTF {##1}
2955            {
2956              \stepcounter{footnote}
2957              \int_gset_eq:Nc \g__enumext_footnote_int { c@footnote }
2958            }
2959            {
2960              \int_gset:Nn \g__enumext_footnote_int { ##1 }
2961            }
2962          \footnotemark [ \g__enumext_footnote_int ]
2963          \seq_gput_right:Nn \g__enumext_footnote_arg_seq { ##2 }
2964          \seq_gput_right:NV \g__enumext_footnote_int_seq \g__enumext_footnote_int
2965        }
2966    }
2967  \cs_new_protected:Nn \__enumext_print_footnote:
2968    {
2969      \seq_if_empty:NF \g__enumext_footnote_int_seq
2970        {
2971          \seq_map_pairwise_function:NNN
2972            \g__enumext_footnote_int_seq
2973            \g__enumext_footnote_arg_seq
2974            \__enumext_footnotetext:nn
2975        }
2976    }
```

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

11.32   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.

11.32.1   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_key_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: (§11.33).

```
2977  \cs_new_protected:Npn \__enumext_default_item:n #1
2978    {
2979      \tl_if_novalue:nTF {#1}
2980        {
2981          \bool_if:NT \l__enumext_check_answers_bool
2982            {
2983              \int_gincr:N \g__enumext_item_number_int
2984              \bool_set_true:N \l__enumext_item_number_bool
2985            }
2986          \bool_set_true:c { l__enumext_wrap_label_ \__enumext_level: _bool }
2987          \__enumext_item_std:w \tl_use:c { l__enumext_fake_item_indent_ \__enumext_level: _tl }
2988        }
2989        {
2990          \bool_set_eq:cc
```

```
2991              { l__enumext_wrap_label_ \__enumext_level: _bool }
2992              { l__enumext_wrap_label_opt_ \__enumext_level: _bool }
2993          \__enumext_item_std:w [#1] \tl_use:c { l__enumext_fake_item_indent_ \__enumext_level: _tl
2994          }
2995      }
```

(*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-pos* or by the second optional argument.

Then we will see the state of the variable \l__enumext_check_ans_key_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: (§11.33).

```
2996  \cs_new_protected:Npn \__enumext_starred_item:nn #1 #2
2997    {
2998      \tl_if_novalue:nF {#1}
2999        {
3000          \tl_set:cn { l__enumext_item_symbol_ \__enumext_level: _tl } {#1}
3001        }
3002      \tl_gset_eq:Nc \g__enumext_item_symbol_tl { l__enumext_item_symbol_ \__enumext_level: _tl }
3003      \tl_if_novalue:nTF {#2}
3004        {
3005          \dim_set_eq:cc
3006            { l__enumext_item_symbol_sep_ \__enumext_level: _dim }
3007            { l__enumext_labelsep_ \__enumext_level: _dim }
3008        }
3009        {
3010          \dim_set:cn { l__enumext_item_symbol_sep_ \__enumext_level: _dim } {#2}
3011        }
3012      \bool_if:NT \l__enumext_check_answers_bool
3013        {
3014          \int_gincr:N \g__enumext_item_number_int
3015          \bool_set_true:N \l__enumext_item_number_bool
3016        }
3017      \bool_set_true:c { l__enumext_wrap_label_ \__enumext_level: _bool }
3018      \__enumext_item_std:w \tl_use:c { l__enumext_fake_item_indent_ \__enumext_level: _tl }
3019    }
```

(*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 (§11.34.2).

```
3020  \cs_new_protected:Nn \__enumext_redefine_item:
3021    {
3022      \RenewDocumentCommand \item { s o o }
3023        {
3024          \bool_if:nTF {##1}
3025            {
3026              \__enumext_starred_item:nn {##2} {##3}
3027            }
3028            { \__enumext_default_item:n {##2} }
3029        }
3030    }
```

(*End of definition for* __enumext_redefine_item:.)

11.32.2 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.

```
3031  \cs_new_protected:Npn \__enumext_keyans_default_item:n #1
3032    {
3033      \tl_if_novalue:nTF { #1 }
3034        {
3035          \bool_set_true:N \l__enumext_wrap_label_v_bool
3036          \__enumext_item_std:w \tl_use:N \l__enumext_fake_item_indent_v_tl
3037        }
3038        {
3039          \bool_set_eq:NN \l__enumext_wrap_label_v_bool \l__enumext_wrap_label_opt_v_bool
3040          \__enumext_item_std:w [#1] \tl_use:N \l__enumext_fake_item_indent_v_tl
3041        }
3042    }
```

(*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⟩.

```
3043  \cs_new_protected:Npn \__enumext_keyans_starred_item:n #1
3044    {
3045      \tl_set_eq:NN \l__enumext_store_current_label_tmp_tl \l__enumext_label_v_tl
3046      \__enumext_keyans_show_left:n { #1 }
3047      \bool_set_true:N \l__enumext_wrap_label_v_bool
3048      \__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⟩.

```
3049      \tl_set_eq:NN \l__enumext_label_v_tl \l__enumext_store_current_label_tmp_tl
3050      \__enumext_keyans_addto_prop:n { #1 }
3051      \__enumext_keyans_store_ref:
3052      \__enumext_keyans_addto_seq:n { #1 }
3053      \int_gincr:N \g__enumext_check_starred_cmd_int
3054    }
```

(*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 (§11.34.2).

```
3055  \cs_new_protected:Nn \__enumext_keyans_redefine_item:
3056    {
3057      \RenewDocumentCommand \item { s o }
3058        {
3059          \bool_if:nTF {##1}
3060            {
3061              \peek_remove_spaces:n
3062                {
3063                  \__enumext_keyans_starred_item:n {##2}
3064                }
3065            }
3066            {
3067              \__enumext_keyans_default_item:n {##2}
3068            }
3069        }
3070    }
```

(*End of definition for* \item* *and* __enumext_keyans_redefine_item:. *This function is documented on page* 14.)

11.33 Redefining \makelabel command

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

11.33.1 Redefining \makelabel for enumext

__enumext_item_starred:

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

```
3071 \cs_new_protected:Nn \__enumext_item_starred:
3072   {
3073     \tl_if_empty:cF { l__enumext_item_symbol_ \__enumext_level: _tl }
3074       {
3075         \mode_leave_vertical:
3076         \skip_horizontal:n { -\dim_use:c { l__enumext_item_symbol_sep_ \__enumext_level: _dim } }
3077         \makebox[ 0pt ][ r ]{ \g__enumext_item_symbol_tl }
3078         \skip_horizontal:n { \dim_use:c { l__enumext_item_symbol_sep_ \__enumext_level: _dim } }
3079       }
3080   }
```

(*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 (§11.34.2).

```
3081 \cs_new_protected:Nn \__enumext_make_label:
3082   {
3083     \RenewDocumentCommand \makelabel { m }
3084       {
3085         \tl_use:c { l__enumext_label_fill_left_ \__enumext_level: _tl }
3086         \tl_use:c { l__enumext_label_font_style_ \__enumext_level: _tl }
3087         \bool_if:cTF { l__enumext_wrap_label_ \__enumext_level: _bool }
3088           {
3089             \__enumext_item_starred:
3090             \use:c { __enumext_wrapper_label_ \__enumext_level: :n } { ##1 }
3091           }
3092           { ##1 }
3093         \tl_use:c { l__enumext_label_fill_right_ \__enumext_level: _tl }
3094         \tl_gclear:N \g__enumext_item_symbol_tl
3095       }
3096   }
```

(*End of definition for* __enumext_make_label:.)

11.33.2 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 (§11.34.2).

```
3097 \cs_new_protected:Nn \__enumext_keyans_make_label:
3098   {
3099     \RenewDocumentCommand \makelabel { m }
3100       {
3101         \tl_use:N \l__enumext_label_fill_left_v_tl
3102         \tl_use:N \l__enumext_label_font_style_v_tl
3103         \bool_if:NTF \l__enumext_wrap_label_v_bool
3104           {
3105             \__enumext_wrapper_label_v:n { ##1 }
3106           }
3107           { ##1 }
3108         \tl_use:N \l__enumext_label_fill_right_v_tl
3109       }
3110   }
```

(*End of definition for* __enumext_keyans_make_label:.)

11.34 Second argument of the lists

At this point of the code we have already programmed most the necessary tools to create a custom list environment, remember that the function __enumext_start_list:nn takes two arguments, the first one we have ready, the second one we will define for all the levels of the environment enumext and the environment keyans.
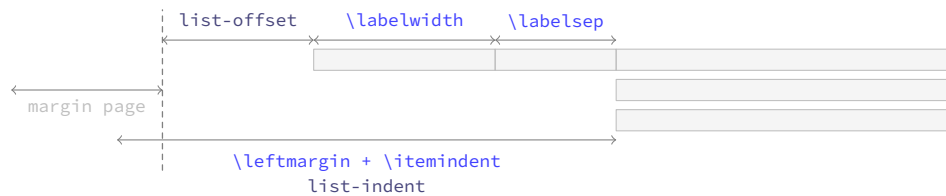
Figure 9: Representation of standard horizontal lengths in `list` environment.

11.34.1 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.
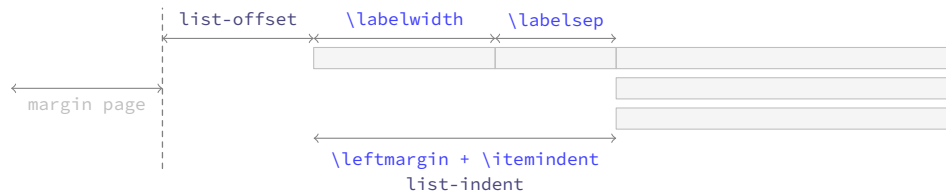
Figure 10: Representation of horizontal lengths concept in list in enumext.

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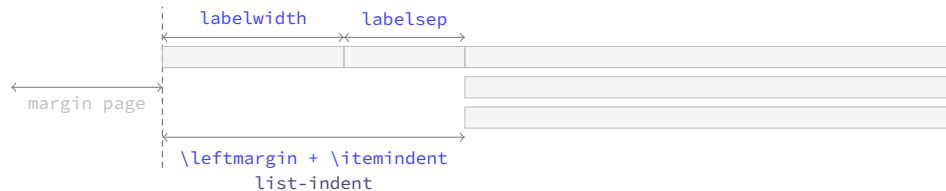
Figure 11: Default horizontal lengths in enumext.

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

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

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

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

This function is passed to `\__enumext_list_arg_two_X:` which is used in the definition of the enumext and keyans environments (§11.34.2).

```
3111  \cs_new_protected:Npn \__enumext_calc_hspace:NNNNNNN #1 #2 #3 #4 #5 #6 #7
3112    {
3113      \dim_compare:nNnT { #1 } < { \c_zero_dim }
3114        {
3115          \msg_warning:nnnV { enumext } { width-non-positive }{ labelwidth }{ #1 }
3116          \dim_set:Nn #1 { \dim_abs:n { #1 } }
3117        }
3118      \dim_compare:nNnT { #2 } < { \c_zero_dim }
3119        {
3120          \msg_warning:nnnV { enumext } { width-negative }{ labelsep }{ #2 }
3121          \dim_set:Nn #2 { \dim_abs:n { #2 } }
3122        }
```

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

```
3123        \bool_if:nF #7 { \dim_set:Nn #4 { #1 + #2} }
```

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

```
3124        \dim_compare:nNnTF { #4 } < { \c_zero_dim }
3125          {
3126            \dim_set:Nn #6 { #1 + #2 - #4}
3127            \dim_set:Nn #5 { #1 + #2 + #3 - #6 }
3128          }
```

```
3129        {
3130            \dim_compare:nNnT { #4 } = { #1 + #2 }
3131              { \dim_set:Nn #6 { \c_zero_dim } }
3132            \dim_compare:nNnT { #4 } < { #1 + #2 }
3133              { \dim_set:Nn #6 { #1 + #2 - #4} }
3134            \dim_compare:nNnT { #4 } > { #1 + #2 }
3135              {
3136                \dim_set:Nn #6 { -#1 - #2 + #4}
3137                \dim_set:Nn #6 { #6*-1}
3138              }
3139            \dim_set:Nn #5 { #1 + #2 + #3 - #6 }
3140          }
3141    }
3142  \cs_generate_variant:Nn \__enumext_calc_hspace:NNNNNNN { ccccccc }
```

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

11.34.2   Setting second argument of the lists

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

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

```
3143  \cs_set_protected:Npn \__enumext_tmp:n #1
3144    {
3145      \cs_new_protected:cpn { __enumext_list_arg_two_#1: }
3146        {
3147          \__enumext_calc_hspace:ccccccc
3148            { l__enumext_labelwidth_#1_dim } { l__enumext_labelsep_#1_dim }
3149            { l__enumext_listoffset_#1_dim } { l__enumext_leftmargin_tmp_#1_dim }
3150            { l__enumext_leftmargin_#1_dim } { l__enumext_itemindent_#1_dim }
3151            { l__enumext_leftmargin_tmp_#1_bool }
3152          \clist_map_inline:nn
3153            { labelsep, labelwidth, itemindent, leftmargin, rightmargin, listparindent }
3154            { \dim_set_eq:cc {####1} { l__enumext_####1_#1_dim } }
3155          \clist_map_inline:nn { topsep, parsep, partopsep, itemsep }
3156            { \skip_set_eq:cc {####1} { l__enumext_####1_#1_skip } }
3157          \usecounter { enumX#1 }
3158          \setcounter { enumX#1 } { \int_eval:n { \int_use:c { l__enumext_start_#1_int } - 1 } }
3159          \str_if_eq:nnTF {#1} { v }
3160            {
3161              \__enumext_keyans_redefine_item:
3162              \__enumext_keyans_make_label:
3163              \__enumext_keyans_ref:
3164              \__enumext_keyans_fake_item:
3165              \bool_if:cT { l__enumext_show_length_#1_bool }
3166                {
3167                  \msg_term:nnnn { enumext } { list-lengths-not-nested } { v } { keyans }
3168                }
3169            }
3170            {
3171              \__enumext_redefine_item:
3172              \__enumext_make_label:
3173              \__enumext_standar_ref:
3174              \__enumext_fake_item:
3175              \bool_if:cT { l__enumext_show_length_#1_bool }
3176                {
3177                  \msg_term:nnne { enumext } { list-lengths } {#1} { \int_use:N \l__enumext_level_i
3178                }
3179            }
3180        }
3181    }
3182  \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.

```
3183  \cs_set_protected:Npn \__enumext_tmp:n #1
3184    {
3185      \cs_new_protected:cpn { __enumext_list_arg_two_#1: }
3186        {
```

```
3187          \__enumext_calc_hspace:ccccccc
3188            { l__enumext_labelwidth_#1_dim } { l__enumext_labelsep_#1_dim }
3189            { l__enumext_listoffset_#1_dim } { l__enumext_leftmargin_tmp_#1_dim }
3190            { l__enumext_leftmargin_#1_dim } { l__enumext_itemindent_#1_dim }
3191            { l__enumext_leftmargin_tmp_#1_bool }
3192          \clist_map_inline:nn
3193            { labelsep, labelwidth, itemindent, leftmargin, rightmargin, listparindent }
3194            { \dim_set_eq:cc {####1} { l__enumext_####1_#1_dim } }
3195          \clist_map_inline:nn { topsep, parsep, partopsep, itemsep }
3196            { \skip_set_eq:cc {####1} { l__enumext_####1_#1_skip } }
3197          \skip_set_eq:Nc \parsep  { l__enumext_itemsep_#1_skip }
3198          \skip_zero:N \partopsep
3199          \usecounter { enumX#1 }
3200          \setcounter { enumX#1 } { \int_eval:n { \int_use:c { l__enumext_start_#1_int } - 1 } }
3201          \__enumext_starred_ref:
3202          \str_if_eq:nnTF {#1} { vii }
3203            {
3204              \__enumext_fake_item_vii:
3205              \bool_if:cT { l__enumext_show_length_vii_bool }
3206                { \msg_term:nnnn { enumext } { list-lengths-not-nested } { vii } { enumext* } }
3207            }
3208            {
3209              \__enumext_fake_item_viii:
3210              \bool_if:cT { l__enumext_show_length_#1_bool }
3211                { \msg_term:nnnn { enumext } { list-lengths-not-nested } { #1 } { keyans* } }
3212            }
3213          }
3214        }
3215  \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:.)

11.35   The environment enumext

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

```
3216  \NewDocumentEnvironment{enumext}{ O{} }
3217    {
3218      \__enumext_safe_exec:
3219      \__enumext_parse_keys:n {#1}
3220      \__enumext_before_list:
3221      \__enumext_start_store_level:
3222      \__enumext_start_list:nn
3223        { \tl_use:c { l__enumext_label_ \__enumext_level: _tl } }
3224        {
3225          \use:c { __enumext_list_arg_two_ \__enumext_level: : }
3226          \__enumext_before_keys_exec:
3227        }
3228      \__enumext_after_args_exec:
3229    }
3230    {
3231      \__enumext_stop_list:
3232      \__enumext_stop_store_level:
3233      \__enumext_after_list:
3234    }
```

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

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

```
3235  \cs_new_protected:Nn \__enumext_safe_exec:
3236    {
3237      \__enumext_internal_mini_page:
3238      \__enumext_is_not_nested:
3239      \int_incr:N \l__enumext_level_int
3240      \int_compare:nNnT { \l__enumext_level_int } > { 4 }
3241        { \msg_fatal:nn { enumext } { list-too-deep } }
3242      \bool_set_true:N \l__enumext_standar_bool
```

```
3243        \__enumext_is_on_first_level:
3244      }
```

(*End of definition for* __enumext_safe_exec:.)

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

```
3245  \cs_new_protected:Npn \__enumext_parse_keys:n #1
3246    {
3247      \tl_if_novalue:nF {#1}
3248        {
3249          \str_clear:N \l__enumext_series_str
3250          \int_compare:nNnTF { \l__enumext_level_int } = { 1 }
3251            {
3252              \keys_set:nn { enumext / level-1 } {#1}
3253              \__enumext_parse_series:n {#1}
3254              \__enumext_nested_base_line_fix:
3255            }
3256            {
3257              \exp_args:Ne \keys_set:nn
3258                { enumext / level-\int_use:N \l__enumext_level_int } {#1}
3259            }
3260          \__enumext_store_active_keys:n {#1}
3261        }
3262    }
```

(*End of definition for* __enumext_parse_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⟩ for the command \anskey and the environment anskey*.

```
3263  \cs_new_protected:Nn \__enumext_start_store_level:
3264    {
3265      \bool_lazy_all:nT
3266        {
3267          { \bool_if_p:N \l__enumext_store_active_bool }
3268          { \bool_not_p:n { \l__enumext_keyans_env_bool } }
3269          { \bool_if_p:N \g__enumext_standar_bool }
3270        }
3271        {
3272          \int_compare:nNnT { \l__enumext_level_int } > { 1 }
3273            {
3274              \bool_set_true:c { l__enumext_store_upper_level_ \__enumext_level: _bool }
3275              \__enumext_store_level_open:
3276            }
3277        }
```

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

```
3278      \bool_lazy_all:nT
3279        {
3280          { \bool_if_p:N \l__enumext_store_active_bool }
3281          { \bool_not_p:n { \l__enumext_keyans_env_bool } }
3282          { \int_compare_p:nNn { \l__enumext_level_h_int } = { 1 } }
3283        }
3284        {
3285          \int_compare:nNnT { \l__enumext_level_int } > { 0 }
3286            {
3287              \bool_set_true:c { l__enumext_store_upper_level_ \__enumext_level: _bool }
3288              \__enumext_store_level_open:
3289            }
3290        }
3291    }
3292  \cs_new_protected:Nn \__enumext_stop_store_level:
3293    {
3294      \bool_if:cT { l__enumext_store_upper_level_ \__enumext_level: _bool }
3295        {
3296          \__enumext_store_level_close:
```

```
3297          }
3298        }
```

(*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.

```
3299  \cs_new_protected:Nn \__enumext_before_list:
3300    {
3301      \__enumext_vspace_above:
3302      \__enumext_before_args_exec:
```

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

```
3303        \__enumext_check_ans_active:
```

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.

```
3304        \dim_compare:nNnT
3305          { \dim_use:c { l__enumext_minipage_right_ \__enumext_level: _dim } } > { \c_zero_dim }
3306          {
3307            \dim_set:cn { l__enumext_minipage_left_ \__enumext_level: _dim }
3308              {
3309                \linewidth
3310                - \dim_use:c { l__enumext_minipage_right_ \__enumext_level: _dim }
3311                - \dim_use:c { l__enumext_minipage_hsep_ \__enumext_level: _dim }
3312              }
```

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*.

```
3313            \bool_set_true:c { l__enumext_minipage_active_ \__enumext_level: _bool }
3314            \int_gincr:N \g__enumext_minipage_stat_int
3315            \__enumext_mini_addvspace:
3316            \nointerlineskip\noindent
3317            \begin{__enumext_mini_env*}
3318              { \dim_use:c { l__enumext_minipage_left_ \__enumext_level: _dim } }
3319          }
3320        \__enumext_multicols_start:
3321      }
```

(*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.

```
3322  \cs_new_protected:Nn \__enumext_multicols_start:
3323    {
3324      \int_compare:nNnT
3325        { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } > { 1 }
3326        {
3327          \dim_compare:nNnT
3328            { \dim_use:c { l__enumext_columns_sep_ \__enumext_level: _dim } } = { \c_zero_dim }
3329            {
3330              \dim_set:cn { l__enumext_columns_sep_ \__enumext_level: _dim }
3331                {
3332                  ( \dim_use:c { l__enumext_labelwidth_ \__enumext_level: _dim }
3333                    + \dim_use:c { l__enumext_labelsep_ \__enumext_level: _dim }
3334                  ) / \int_use:c { l__enumext_columns_ \__enumext_level: _int }
3335                  - \dim_use:c { l__enumext_listoffset_ \__enumext_level: _dim }
3336                }
3337            }
```

```
3338            \dim_set_eq:Nc \columnsep { l__enumext_columns_sep_ \__enumext_level: _dim  }
3339            \skip_zero:N \multicolsep
3340            \int_compare:nNnT { \l__enumext_level_int } > { 1 }
3341              {
3342                \dim_zero:N \columnseprule
3343              }
```

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.

```
3344            \bool_if:cF { l__enumext_minipage_active_ \__enumext_level: _bool }
3345              {
3346                \__enumext_multi_addvspace:
3347              }
3348            \raggedcolumns
3349            \begin{multicols}{ \int_use:c { l__enumext_columns_ \__enumext_level: _int } }
3350          }
3351      }
```

(*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.

```
3352  \cs_new_protected:Nn \__enumext_multicols_stop:
3353    {
3354      \int_compare:nNnT
3355        { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } > { 1 }
3356        {
3357          \end{multicols}
3358          \bool_if:cF { l__enumext_minipage_active_ \__enumext_level: _bool }
3359            {
3360              \par\addvspace{ \skip_use:c { l__enumext_multicols_below_ \__enumext_level: _skip } }
3361            }
3362        }
3363    }
```

(*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.

```
3364  \cs_new_protected:Nn \__enumext_after_list:
3365    {
3366      \bool_if:cTF { l__enumext_minipage_active_ \__enumext_level: _bool }
3367        {
3368          \int_compare:nNnT { \g__enumext_minipage_stat_int } = { 1 }
3369            {
3370              \msg_warning:nn { enumext } { missing-miniright }
3371              \miniright
3372            }
3373          \int_gzero:N \g__enumext_minipage_stat_int
3374          \end{__enumext_mini_env*}
3375          \par\addvspace { \l__enumext_minipage_after_skip }
3376        }
3377        { \__enumext_multicols_stop: }
```

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

```
3378      \__enumext_check_ans_key_hook:
```

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

```
3379      \__enumext_after_stop_list:
3380      \__enumext_vspace_below:
3381      \bool_set_false:N \l__enumext_standar_bool
3382      \__enumext_resume_save_counter:
3383    }
```

(*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.

```
3384 \__enumext_after_env:nn {enumext} { \__enumext_execute_after_env: }
```

11.36 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.

```
3385 \NewDocumentEnvironment{keyans}{ O{} }
3386   {
3387     \__enumext_keyans_safe_exec:
3388     \__enumext_keyans_parse_keys:n {#1}
3389     \__enumext_before_list_v:
3390     \__enumext_start_list:nn
3391       { \tl_use:N \l__enumext_label_v_tl }
3392       {
3393         \__enumext_list_arg_two_v:
3394         \__enumext_before_keys_exec_v:
3395       }
3396     \__enumext_after_args_exec_v:
3397   }
3398   {
3399     \__enumext_check_starred_cmd:n { item }
3400     \__enumext_stop_list:
3401     \__enumext_after_list_v:
3402   }
```

(*End of definition for* keyans. *This function is documented on page* *13.*)

__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.

```
3403 \cs_new_protected:Nn \__enumext_keyans_safe_exec:
3404   {
3405     \bool_if:NF \l__enumext_store_active_bool
3406       {
3407         \msg_error:nnnn { enumext } { wrong-place }{ keyans }{ save-ans }
3408       }
3409     \int_incr:N \l__enumext_keyans_level_int
3410     \bool_set_true:N \l__enumext_keyans_env_bool
3411     \__enumext_keyans_start_line:
3412     % Set false for interfering with enumext nested in keyans (yes, its possible and crayze)
3413     \bool_set_false:N \l__enumext_store_active_bool
3414     \int_compare:nNnT { \l__enumext_keyans_level_int } > { 1 }
3415       {
3416         \msg_error:nn { enumext } { keyans-nested }
3417       }
3418     \int_compare:nNnT { \l__enumext_level_int } > { 1 }
3419       {
3420         \msg_error:nn { enumext } { keyans-wrong-level }
3421       }
3422   }
```

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

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

```
3423 \cs_new_protected:Npn \__enumext_keyans_parse_keys:n #1
3424   {
3425     \keys_set:nn { enumext / keyans } {#1}
3426   }
```

(*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.

```
3427  \cs_new_protected:Nn \__enumext_before_list_v:
3428    {
3429      \__enumext_vspace_above_v:
3430      \__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.

```
3431      \dim_compare:nNnT { \l__enumext_minipage_right_v_dim } > { \c_zero_dim }
3432        {
3433          \dim_set:Nn \l__enumext_minipage_left_v_dim
3434            {
3435              \linewidth - \l__enumext_minipage_right_v_dim - \l__enumext_minipage_hsep_v_dim
3436            }
```

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*.

```
3437          \bool_set_true:N \l__enumext_minipage_active_v_bool
3438          \int_gincr:N \g__enumext_minipage_stat_int
3439          \__enumext_keyans_mini_addvspace:
3440          \nointerlineskip\noindent
3441          \begin{__enumext_mini_env*}{ \l__enumext_minipage_left_v_dim }
3442        }
```

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

```
3443      \__enumext_keyans_multicols_start:
3444    }
```

(*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.

```
3445  \cs_new_protected:Nn \__enumext_keyans_multicols_start:
3446    {
3447      \int_compare:nNnT { \l__enumext_columns_v_int } > { 1 }
3448        {
```

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

```
3449          \dim_compare:nNnT { \l__enumext_columns_sep_v_dim } = { \c_zero_dim }
3450            {
3451              \dim_set:Nn \l__enumext_columns_sep_v_dim
3452                {
3453                  (
3454                    \l__enumext_labelwidth_v_dim + \l__enumext_labelsep_v_dim
3455                  ) / \l__enumext_columns_v_int
3456                  - \l__enumext_listoffset_v_dim
3457                }
3458            }
3459          \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).

```
3460          \skip_zero:N \multicolsep
3461          \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.

```
3462          \bool_if:NF \l__enumext_minipage_active_v_bool
3463            {
3464              \__enumext_keyans_multi_addvspace:
3465            }
3466          \raggedcolumns
```

```
3467          \begin{multicols}{ \l__enumext_columns_v_int }
3468        }
3469    }
```

(*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.

```
3470  \cs_new_protected:Nn \__enumext_keyans_multicols_stop:
3471    {
3472      \int_compare:nNnT { \l__enumext_columns_v_int } > { 1 }
3473        {
3474          \end{multicols}
3475          \bool_if:NF \l__enumext_minipage_active_v_bool
3476            {
3477              \par\addvspace{ \l__enumext_multicols_below_v_skip }
3478            }
3479        }
3480    }
```

(*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.

```
3481  \cs_new_protected:Nn \__enumext_after_list_v:
3482    {
3483      \bool_if:NTF \l__enumext_minipage_active_v_bool
3484        {
3485          \int_compare:nNnT { \g__enumext_minipage_stat_int } = { 1 }
3486            {
3487              \msg_warning:nn { enumext } { missing-miniright }
3488              \miniright
3489            }
3490          \int_gzero:N \g__enumext_minipage_stat_int
3491          \end{__enumext_mini_env*}
3492          \par\addvspace{ \l__enumext_minipage_after_skip }
3493        }
3494        { \__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.

```
3495      \bool_set_false:N \l__enumext_keyans_env_bool
3496      \__enumext_after_stop_list_v:
3497      \__enumext_vspace_below_v:
3498    }
```

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

11.37 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.

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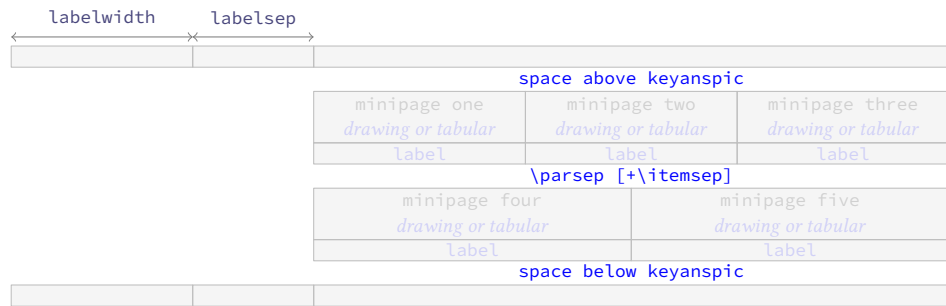
Figure 12: Representation of the keyanspic spacing in enumext.

11.37.1 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.

```
3499  \NewDocumentCommand \anspic { s o +m }
3500    {
```

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

```
3501      \bool_if:NF \l__enumext_store_active_bool
3502        {
3503          \msg_error:nnnn { enumext } { wrong-place }{ keyanspic }{ save-ans }
3504        }
3505      \int_compare:nNnT { \l__enumext_level_int } > { 1 }
3506        {
3507          \msg_error:nn { enumext } { keyanspic-wrong-level }
3508        }
3509      \int_compare:nNnT { \l__enumext_keyans_level_int } = { 1 }
3510        {
3511          \msg_error:nnnn { enumext } { command-wrong-place }{ anspic }{ keyans }
3512        }
```

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.

```
3513      \seq_put_right:Nn \l__enumext_keyans_pic_body_seq
3514        {
3515          \__enumext_keyans_anspic_code:nnn { #1 } { #2 } { #3 }
3516        }
3517    }
```

(*End of definition for* \anspic. *This function is documented on page* 15.)

__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.

```
3518  \cs_new_protected:Nn \__enumext_keyans_anspic_code:nnn
3519    {
3520      \stepcounter { enumXvi }
3521      #3 \\
3522      \bool_if:nT { #1 }
3523        {
3524          \__enumext_keyans_addto_prop:n { #2 }
3525          \__enumext_keyans_store_ref:
3526          \__enumext_keyans_addto_seq:n { #2 }
3527          \int_gincr:N \g__enumext_check_starred_cmd_int
3528          \bool_lazy_or:nnT
3529            { \bool_if_p:N \l__enumext_show_answer_bool }
3530            { \bool_if_p:N \l__enumext_show_position_bool }
3531            {
3532              \tl_set_eq:NN \l__enumext_label_v_tl \l__enumext_label_vi_tl
3533              \__enumext_keyans_show_left:n { #2 }
3534              \tl_set_eq:NN \l__enumext_label_vi_tl \l__enumext_label_v_tl
3535            }
3536        }
3537      \tl_use:N \l__enumext_label_font_style_v_tl
3538      \__enumext_wrapper_label_v:n { \l__enumext_label_vi_tl } \__enumext_keyans_show_item_opt:
3539    }
```

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

11.37.2   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.

```
3540  \NewDocumentEnvironment{keyanspic}{ o }
3541    {
3542      \__enumext_keyans_pic_safe_exec:
3543      \__enumext_start_list:nn
3544        { }
3545        {
3546            \__enumext_keyans_pic_arg_two:
3547        }
```

We apply the "adjusted" vertical spacing above the environment

```
3548        \vspace { \l__enumext_keyans_pic_above_skip }
3549    }
```

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.

```
3550    {
3551      \tl_if_novalue:nTF { #1 }
3552        {
3553            \__enumext_keyans_pic_do:e { \seq_count:N \l__enumext_keyans_pic_body_seq }
3554        }
3555        { \__enumext_keyans_pic_do:n { #1 } }
3556      \__enumext_stop_list:
3557      \__enumext_check_starred_cmd:n { anspic }
3558      \setcounter { enumXvi } { 0 }
3559      \vspace { \l__enumext_topsep_v_skip }
3560      %\bool_set_false:N \l__enumext_store_active_bool
3561    }
```

(*End of definition for* keyanspic. *This function is documented on page* 14.)

__enumext_keyans_pic_safe_exec:

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

```
3562  \cs_new_protected:Nn \__enumext_keyans_pic_safe_exec:
3563    {
3564      \int_incr:N \l__enumext_keyans_pic_level_int
3565      \int_compare:nNnT { \l__enumext_keyans_pic_level_int } > { 1 }
3566        {
3567            \msg_error:nn { enumext } { keyanspic-nested }
3568        }
3569      \__enumext_keyans_start_line:
3570    }
```

(*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.

```
3571  \cs_new_protected:Npn \__enumext_keyans_pic_skip_abs:N #1
3572    {
3573      \dim_compare:nNnT { #1 } < { 0pt }
3574        { \skip_set:Nn #1 { -#1 } }
3575    }
```

(*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.

```
3576  \cs_new_protected:Nn \__enumext_keyans_pic_arg_two:
3577    {
```

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.

```
3578      \bool_set_false:N \l__enumext_leftmargin_tmp_v_bool
3579      \__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.

```
3580      \skip_add:Nn \parsep { \itemsep }
3581      \dim_add:Nn  \leftmargin { -\labelwidth - \labelsep }
3582      \dim_zero:N  \labelwidth
3583      \dim_zero:N  \listparindent
3584      \dim_zero:N  \labelsep
3585      \skip_zero:N \partopsep
3586      \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.

```
3587      \__enumext_keyans_pic_skip_abs:N \parsep
3588      \skip_set:Nn \l__enumext_keyans_pic_above_skip
3589        {
3590          \box_dp:N \strutbox
3591          + \l__enumext_topsep_v_skip
3592          - \parsep
3593        }
3594      \__enumext_item_std:w \scan_stop:
3595    }
```

(*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`.

```
3596  \cs_new_protected:Nn \__enumext_keyans_pic_do:n
3597    {
3598      \clist_map_function:nN { #1 } \__enumext_keyans_pic_row:n
3599    }
3600  \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.

```
3601  \cs_new_protected:Nn \__enumext_keyans_pic_row:n
3602    {
3603      \dim_set:Nn \l__enumext_keyans_pic_width_dim { \linewidth / #1 }
3604      \int_set:Nn \l__enumext_keyans_pic_above_int { \l__enumext_keyans_pic_below_int }
3605      \int_set:Nn \l__enumext_keyans_pic_below_int { \l__enumext_keyans_pic_above_int + #1 }
3606      \int_step_inline:nnn
3607        { \l__enumext_keyans_pic_above_int + 1 }
3608        { \l__enumext_keyans_pic_below_int }
3609        {
3610          \__enumext_minipage:w [ b ]{ \l__enumext_keyans_pic_width_dim }
3611            \centering
3612            \seq_item:Nn \l__enumext_keyans_pic_body_seq { ##1 }
3613          \__enumext_endminipage:
3614        }
3615      \par
3616    }
```

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

11.38   The horizontal environments

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*⟩).

11.38.1 Functions for item box width

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

```
3617  \cs_new_protected:Nn \__enumext_starred_columns_set_vii:
3618    {
3619      \dim_compare:nNnT { \l__enumext_columns_sep_vii_dim } = { \c_zero_dim }
3620        {
3621          \dim_set:Nn \l__enumext_columns_sep_vii_dim
3622            {
3623              ( \l__enumext_labelwidth_vii_dim + \l__enumext_labelsep_vii_dim )
3624              / \l__enumext_columns_vii_int
3625            }
3626        }
3627      \int_set:Nn \l__enumext_tmpa_vii_int { \l__enumext_columns_vii_int - 1 }
3628      \dim_set:Nn \l__enumext_item_width_vii_dim
3629        {
3630          ( \linewidth - \l__enumext_columns_sep_vii_dim * \l__enumext_tmpa_vii_int )
3631          / \l__enumext_columns_vii_int - \l__enumext_labelwidth_vii_dim
3632          - \l__enumext_labelsep_vii_dim
3633        }
3634      \dim_zero_new:N \itemwidth
3635    }
3636  \cs_new_protected:Nn \__enumext_starred_columns_set_viii:
3637    {
3638      \dim_compare:nNnT { \l__enumext_columns_sep_viii_dim } = { \c_zero_dim }
3639        {
3640          \dim_set:Nn \l__enumext_columns_sep_viii_dim
3641            {
3642              ( \l__enumext_labelwidth_viii_dim + \l__enumext_labelsep_viii_dim )
3643              / \l__enumext_columns_viii_int
3644            }
3645        }
3646      \int_set:Nn \l__enumext_tmpa_viii_int { \l__enumext_columns_viii_int - 1 }
3647      \dim_set:Nn \l__enumext_item_width_viii_dim
3648        {
3649          ( \linewidth - \l__enumext_columns_sep_viii_dim * \l__enumext_tmpa_viii_int )
3650          / \l__enumext_columns_viii_int - \l__enumext_labelwidth_viii_dim
3651          - \l__enumext_labelsep_viii_dim
3652        }
3653      \dim_zero_new:N \itemwidth
3654    }
```

(*End of definition for* `\__enumext_starred_columns_set_vii:` *and* `\__enumext_starred_columns_set_viii:`.)

11.38.2 Functions for join item columns

The functions `\__enumext_starred_joined_item_vii:n` and `\__enumext_starred_joined_item_-viii:n` will set the *width* of the box in which the content passed to `\item`(⟨*columns*⟩) will be stored together with the value of `\itemwidth`.

```
3655  \cs_new_protected:Npn \__enumext_starred_joined_item_vii:n #1
3656    {
3657      \int_set:Nn \l__enumext_joined_item_vii_int {#1}
3658      \int_compare:nNnT { \l__enumext_joined_item_vii_int } > { \l__enumext_columns_vii_int }
3659        {
3660          \msg_warning:nnee { enumext } { item-joined }
3661            { \int_use:N \l__enumext_joined_item_vii_int }
3662            { \int_use:N \l__enumext_columns_vii_int }
3663          \int_set:Nn \l__enumext_joined_item_vii_int
3664            {
3665              \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + 1
3666            }
3667        }
3668      \int_compare:nNnT
3669        { \l__enumext_joined_item_vii_int }
3670        >
3671        { \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + 1 }
3672        {
3673          \msg_warning:nnee { enumext } { item-joined-columns }
3674            { \int_use:N \l__enumext_joined_item_vii_int }
3675            {
3676              \int_eval:n
```

```
3677                    { \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + 1 }
3678                  }
3679              \int_set:Nn \l__enumext_joined_item_vii_int
3680                {
3681                  \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + 1
3682                }
3683          }
3684      \int_compare:nNnTF { \l__enumext_joined_item_vii_int } > { 1 }
3685        {
3686          \int_set_eq:NN \l__enumext_joined_item_aux_vii_int \l__enumext_joined_item_vii_int
3687          \int_decr:N \l__enumext_joined_item_aux_vii_int
3688          \int_add:Nn \l__enumext_item_column_pos_vii_int { \l__enumext_joined_item_aux_vii_int }
3689          \int_gadd:Nn \g__enumext_item_count_all_vii_int { \l__enumext_joined_item_aux_vii_int }
3690          \dim_set:Nn \l__enumext_joined_width_vii_dim
3691            {
3692              \l__enumext_item_width_vii_dim * \l__enumext_joined_item_vii_int
3693              + (  \l__enumext_labelwidth_vii_dim + \l__enumext_labelsep_vii_dim
3694                 + \l__enumext_columns_sep_vii_dim
3695              )*\l__enumext_joined_item_aux_vii_int
3696            }
3697          \dim_set_eq:NN \itemwidth \l__enumext_joined_width_vii_dim
3698        }
3699        {
3700          \dim_set_eq:NN \l__enumext_joined_width_vii_dim \l__enumext_item_width_vii_dim
3701          \dim_set_eq:NN \itemwidth \l__enumext_item_width_vii_dim
3702        }
3703  }
3704  \cs_new_protected:Npn \__enumext_starred_joined_item_viii:n #1
3705    {
3706      \int_set:Nn \l__enumext_joined_item_viii_int {#1}
3707      \int_compare:nNnT { \l__enumext_joined_item_viii_int } > { \l__enumext_columns_viii_int }
3708        {
3709          \msg_warning:nnee { enumext } { item-joined }
3710            { \int_use:N \l__enumext_joined_item_viii_int }
3711            { \int_use:N \l__enumext_columns_viii_int }
3712          \int_set:Nn \l__enumext_joined_item_viii_int
3713            {
3714              \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1
3715            }
3716        }
3717      \int_compare:nNnT
3718        { \l__enumext_joined_item_viii_int }
3719          >
3720        { \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1 }
3721        {
3722          \msg_warning:nnee { enumext } { item-joined-columns }
3723            { \int_use:N \l__enumext_joined_item_viii_int }
3724            {
3725              \int_eval:n
3726                { \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1 }
3727            }
3728          \int_set:Nn \l__enumext_joined_item_viii_int
3729            {
3730              \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1
3731            }
3732        }
3733      \int_compare:nNnTF { \l__enumext_joined_item_viii_int } > { 1 }
3734        {
3735          \int_set_eq:NN \l__enumext_joined_item_aux_viii_int \l__enumext_joined_item_viii_int
3736          \int_decr:N \l__enumext_joined_item_aux_viii_int
3737          \int_add:Nn \l__enumext_item_column_pos_viii_int { \l__enumext_joined_item_aux_viii_int }
3738          \int_gadd:Nn \g__enumext_item_count_all_viii_int { \l__enumext_joined_item_aux_viii_int }
3739          \dim_set:Nn \l__enumext_joined_width_viii_dim
3740            {
3741              \l__enumext_item_width_viii_dim * \l__enumext_joined_item_viii_int
3742              + (  \l__enumext_labelwidth_viii_dim + \l__enumext_labelsep_viii_dim
3743                 + \l__enumext_columns_sep_viii_dim
3744              )*\l__enumext_joined_item_aux_viii_int
3745            }
3746          \dim_set_eq:NN \itemwidth \l__enumext_joined_width_viii_dim
3747        }
```

```
3748        {
3749            \dim_set_eq:NN \l__enumext_joined_width_viii_dim \l__enumext_item_width_viii_dim
3750            \dim_set_eq:NN \itemwidth \l__enumext_item_width_viii_dim
3751        }
3752    }
```

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

11.38.3 Functions for mini-env, mini-right and mini-right* keys

__enumext_start_mini_vii:
__enumext_stop_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.

```
3753 \cs_new_protected:Nn \__enumext_start_mini_vii:
3754    {
3755        \dim_compare:nNnT { \l__enumext_minipage_right_vii_dim } > { \c_zero_dim }
3756            {
3757                \dim_set:Nn \l__enumext_minipage_left_vii_dim
3758                    {
3759                        \linewidth
3760                        - \l__enumext_minipage_right_vii_dim
3761                        - \l__enumext_minipage_hsep_vii_dim
3762                    }
3763                \bool_set_true:N \l__enumext_minipage_active_vii_bool
3764                \dim_gset_eq:NN
3765                    \g__enumext_minipage_right_vii_dim
3766                    \l__enumext_minipage_right_vii_dim
3767                \__enumext_mini_addvspace_vii:
3768                \nointerlineskip\noindent
3769                \begin{__enumext_mini_env*}{ \l__enumext_minipage_left_vii_dim }
3770            }
3771    }
```

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_env:nn to execute the __enumext_mini_-env* on the *"right side"*.

```
3772 \cs_new_protected:Nn \__enumext_stop_mini_vii:
3773    {
3774        \bool_if:NT \l__enumext_minipage_active_vii_bool
3775            {
3776                \end{__enumext_mini_env*}
3777                \hfill
3778                \bool_gset_true:N \g__enumext_minipage_active_vii_bool
3779            }
3780    }
```

Finally we execute the {⟨*code*⟩} passed to the mini-right or mini-right* keys stored in the variable \g__enumext_miniright_code_vii_tl in the __enumext_mini_env* environment on the *"right side"*. For compatibility with the caption package and possibly other {⟨*code*⟩} passed to this key, we will pass it to a box and then print it.

```
3781 \__enumext_after_env:nn {enumext*}
3782    {
3783        \bool_if:NT \g__enumext_minipage_active_vii_bool
3784            {
3785                \begin{__enumext_mini_env*}{ \g__enumext_minipage_right_vii_dim }
3786                    \par\addvspace { \g__enumext_minipage_right_skip }
3787                    \bool_if:NF \g__enumext_minipage_center_vii_bool
3788                        {
3789                            \tl_put_left:Nn \g__enumext_miniright_code_vii_tl
3790                                {
3791                                    \centering
3792                                }
3793                        }
3794                    \vbox_set_top:Nn \l__enumext_miniright_code_vii_box
3795                        {
3796                            \tl_use:N \g__enumext_miniright_code_vii_tl
3797                        }
3798                    \box_use_drop:N \l__enumext_miniright_code_vii_box
3799                \end{__enumext_mini_env*}
3800                \par\addvspace{ \g__enumext_minipage_after_skip }
```

```
3801        }
3802      \bool_gset_false:N \g__enumext_minipage_active_vii_bool
3803      \bool_gset_true:N \g__enumext_minipage_center_vii_bool
3804      \tl_gclear:N \g__enumext_miniright_code_vii_tl
3805      \dim_gzero:N \g__enumext_minipage_right_vii_dim
3806      \bool_gset_false:N \g__enumext_starred_bool
3807    }
```

(*End of definition for* __enumext_start_mini_vii: *and* __enumext_stop_mini_vii:.)

__enumext_start_mini_viii: The implementation of the `mini-env`, `mini-right` and `mini-right*` keys is identical to the one used in
__enumext_stop_mini_viii: the `enumext*` environment.

```
3808 \cs_new_protected:Nn \__enumext_start_mini_viii:
3809    {
3810      \dim_compare:nNnT { \l__enumext_minipage_right_viii_dim } > { \c_zero_dim }
3811        {
3812          \dim_set:Nn \l__enumext_minipage_left_viii_dim
3813            {
3814              \linewidth
3815              - \l__enumext_minipage_right_viii_dim
3816              - \l__enumext_minipage_hsep_viii_dim
3817            }
3818          \bool_set_true:N \l__enumext_minipage_active_viii_bool
3819          \dim_gset_eq:NN
3820            \g__enumext_minipage_right_viii_dim
3821            \l__enumext_minipage_right_viii_dim
3822          \__enumext_mini_addvspace_viii:
3823          \nointerlineskip\noindent
3824          \begin{__enumext_mini_env*}{ \l__enumext_minipage_left_viii_dim }
3825        }
3826    }
3827 \cs_new_protected:Nn \__enumext_stop_mini_viii:
3828    {
3829      \bool_if:NT \l__enumext_minipage_active_viii_bool
3830        {
3831          \end{__enumext_mini_env*}
3832          \hfill
3833          \bool_gset_true:N \g__enumext_minipage_active_viii_bool
3834        }
3835    }
3836 \__enumext_after_env:nn {keyans*}
3837    {
3838      \bool_if:NT \g__enumext_minipage_active_viii_bool
3839        {
3840          \begin{__enumext_mini_env*}{ \g__enumext_minipage_right_viii_dim }
3841            \par\addvspace { \g__enumext_minipage_right_skip }
3842            \bool_if:NF \g__enumext_minipage_center_viii_bool
3843              {
3844                \tl_put_left:Nn \g__enumext_miniright_code_viii_tl
3845                  {
3846                    \centering
3847                  }
3848              }
3849            \vbox_set_top:Nn \l__enumext_miniright_code_viii_box
3850              {
3851                \tl_use:N \g__enumext_miniright_code_viii_tl
3852              }
3853            \box_use_drop:N \l__enumext_miniright_code_viii_box
3854          \end{__enumext_mini_env*}
3855          \par\addvspace{ \g__enumext_minipage_after_skip }
3856        }
3857      \bool_gset_false:N \g__enumext_minipage_active_viii_bool
3858      \bool_gset_true:N \g__enumext_minipage_center_viii_bool
3859      \tl_gclear:N \g__enumext_miniright_code_viii_tl
3860      \dim_gzero:N \g__enumext_minipage_right_viii_dim
3861    }
```

(*End of definition for* __enumext_start_mini_viii: *and* __enumext_stop_mini_viii:.)

11.39 The environment enumext*

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.

```
3862  \NewDocumentEnvironment{enumext*}{ o }
3863    {
3864      \__enumext_safe_exec_vii:
3865      \__enumext_parse_keys_vii:n {#1}
3866      \__enumext_starred_columns_set_vii:
3867      \__enumext_before_list_vii:
3868      \__enumext_start_store_level_vii:
3869      \__enumext_start_list:nn { }
3870        {
3871          \__enumext_list_arg_two_vii:
3872          \__enumext_before_keys_exec_vii:
3873        }
3874      \item[] \scan_stop:
3875      \cs_set_eq:NN \__enumext_stop_item_tmp_vii: \noindent
3876      \cs_set_eq:NN \item \__enumext_start_item_tmp_vii:
3877    }
3878    {
3879      \__enumext_stop_item_tmp_vii:
3880      \__enumext_remove_extra_parsep_vii:
3881      \__enumext_stop_list:
3882      \__enumext_stop_store_level_vii:
3883      \__enumext_after_list_vii:
3884    }
```

(*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.

```
3885  \cs_new_protected:Nn \__enumext_safe_exec_vii:
3886    {
3887      \__enumext_internal_mini_page:
3888      \__enumext_is_not_nested:
3889      \int_incr:N \l__enumext_level_h_int
3890      \int_compare:nNnT { \l__enumext_level_h_int } > { 1 }
3891        {
3892          \msg_error:nn { enumext } { nested }
3893        }
3894      \bool_set_true:N \l__enumext_starred_bool
3895      \__enumext_is_on_first_level:
3896    }
```

(*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 functions __enumext_parse_serie:n and __enumext_store_active_keys_vii:n and reprocess the ⟨*keys*⟩ to pass them to the storage ⟨*sequence*⟩.

```
3897  \cs_new_protected:Npn \__enumext_parse_keys_vii:n #1
3898    {
3899      \tl_if_novalue:nF {#1}
3900        {
3901          \str_clear:N \l__enumext_series_str
3902          \keys_set:nn { enumext / enumext* } {#1}
3903          \__enumext_parse_series:n {#1}
3904          \__enumext_store_active_keys_vii:n {#1}
3905          \__enumext_nested_base_line_fix:
3906        }
3907    }
```

(*End of definition for* __enumext_parse_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.

```
3908  \cs_new_protected:Nn \__enumext_before_list_vii:
3909    {
3910      \__enumext_vspace_above_vii:
```

```
3911        \__enumext_check_ans_active:
3912        \__enumext_before_args_exec_vii:
3913        \__enumext_start_mini_vii:
3914      }
```

(*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.

```
3915  \cs_new_protected:Nn \__enumext_after_list_vii:
3916    {
3917      \__enumext_stop_mini_vii:
3918      \__enumext_after_stop_list_vii:
3919      \__enumext_check_ans_key_hook:
3920      \__enumext_vspace_below_vii:
3921      \bool_set_false:N \l__enumext_starred_bool
3922      \__enumext_resume_save_counter:
3923    }
```

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

__enumext_start_store_level_vii:
__enumext_stop_store_level_vii:

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

```
3924  \cs_new_protected:Nn \__enumext_start_store_level_vii:
3925    {
3926      \bool_if:NT \l__enumext_store_active_bool
3927        {
3928          \int_compare:nNnT { \l__enumext_level_int } > { 0 }
3929            {
3930              \__enumext_store_level_open_vii:
3931            }
3932        }
3933    }
3934  \cs_new_protected:Nn \__enumext_stop_store_level_vii:
3935    {
3936      \bool_if:NT \l__enumext_store_active_bool
3937        {
3938          \int_compare:nNnT { \l__enumext_level_int } > { 0 }
3939            {
3940              \__enumext_store_level_close_vii:
3941            }
3942        }
3943    }
```

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

11.39.1 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.

```
3944  \cs_new_protected_nopar:Nn \__enumext_start_item_tmp_vii:
3945    {
3946      \__enumext_stop_item_tmp_vii:
3947      \int_incr:N \l__enumext_item_column_pos_vii_int
3948      \int_gincr:N \g__enumext_item_count_all_vii_int
3949      \__enumext_item_peek_args_vii:
3950    }
```

(*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).

```
3951 \cs_new_protected:Nn \__enumext_item_peek_args_vii:
3952   {
3953     \peek_meaning:NTF (
3954       { \__enumext_joined_item_vii:w }
3955       { \__enumext_joined_item_vii:w (1) }
3956   }
```

(*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_standar_item_vii:w`.

```
3957 \cs_new_protected:Npn \__enumext_joined_item_vii:w (#1)
3958   {
3959     \__enumext_starred_joined_item_vii:n {#1}
3960     \peek_meaning_remove:NTF *
3961       { \__enumext_starred_item_vii:w  }
3962       { \__enumext_standar_item_vii:w }
3963   }
```

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

__enumext_standar_item_vii:w

The function `\__enumext_standar_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 ]`.

```
3964 \cs_new_protected:Npn \__enumext_standar_item_vii:w
3965   {
3966     \bool_set_false:N \l__enumext_item_starred_vii_bool
3967       \peek_meaning:NTF [
3968         {
3969           \bool_set_eq:NN
3970             \l__enumext_wrap_label_vii_bool
3971             \l__enumext_wrap_label_opt_vii_bool
3972           \__enumext_start_item_vii:w
3973         }
3974         {
3975           \bool_set_true:N \l__enumext_wrap_label_vii_bool
3976           \legacy_if_set_true:n { @noitemarg }
3977           \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
3978         }
3979   }
```

(*End of definition for* `\__enumext_standar_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*⟩`]`.

```
3980 \cs_new_protected:Npn \__enumext_starred_item_vii:w
3981   {
3982     \bool_set_true:N \l__enumext_item_starred_vii_bool
3983     \bool_set_true:N \l__enumext_wrap_label_vii_bool
3984     \peek_meaning:NTF [
3985       { \__enumext_starred_item_vii_aux_i:w }
3986       { \__enumext_starred_item_vii_aux_ii:w }
3987   }
3988 \cs_new_protected:Npn \__enumext_starred_item_vii_aux_i:w [#1]
3989   {
3990     \tl_gset:Nn \g__enumext_item_symbol_aux_vii_tl {#1}
3991     \__enumext_starred_item_vii_aux_ii:w
3992   }
3993 \cs_new_protected:Npn \__enumext_starred_item_vii_aux_ii:w
```

```
3994      {
3995        \peek_meaning:NTF [
3996          { \__enumext_starred_item_vii_aux_iii:w }
3997          {
3998            \dim_set_eq:NN
3999              \l__enumext_item_symbol_sep_vii_dim
4000              \l__enumext_labelsep_vii_dim
4001            \legacy_if_set_true:n { @noitemarg }
4002            \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
4003          }
4004      }
4005  \cs_new_protected:Npn \__enumext_starred_item_vii_aux_iii:w [#1]
4006      {
4007        \dim_set:Nn \l__enumext_item_symbol_sep_vii_dim {#1}
4008        \legacy_if_set_true:n { @noitemarg }
4009        \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
4010      }
```

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

11.39.2   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 __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.

```
4011  \cs_new_protected_nopar:Npn \__enumext_start_item_vii:w [#1]
4012      {
4013        \cs_set_eq:NN \__enumext_stop_item_tmp_vii: \__enumext_stop_item_vii:
4014        \legacy_if:nT { @noitemarg }
4015          {
4016            \legacy_if_set_false:n { @noitemarg }
4017            \legacy_if:nT { @nmbrlist }
4018              {
4019                \bool_if:NT \l__enumext_hyperref_bool
4020                  {
4021                    \legacy_if_set_true:n { @hyper@item }
4022                  }
4023                \refstepcounter{enumXvii}
4024                \bool_if:NT \l__enumext_check_answers_bool
4025                  {
4026                    \int_gincr:N \g__enumext_item_number_int
4027                    \bool_set_true:N \l__enumext_item_number_bool
4028                  }
4029              }
4030          }
```

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.

```
4031        \group_begin:
4032          \lrbox{ \l__enumext_item_text_vii_box }
4033            \bool_if:NF \l__enumext_footnotes_key_bool
4034              {
4035                \__enumext_renew_footnote:
4036              }
4037            \bool_if:NT \l__enumext_item_starred_vii_bool
4038              {
4039                \tl_if_blank:VT \g__enumext_item_symbol_aux_vii_tl
4040                  {
4041                    \tl_gset_eq:NN
4042                      \g__enumext_item_symbol_aux_vii_tl \l__enumext_item_symbol_vii_tl
4043                  }
4044                \mode_leave_vertical:
```

```
4045          \skip_horizontal:n { -\l__enumext_item_symbol_sep_vii_dim }
4046          \makebox[ 0pt ][ r ]{ \g__enumext_item_symbol_aux_vii_tl }
4047          \skip_horizontal:N \l__enumext_item_symbol_sep_vii_dim
4048          \tl_gclear:N \g__enumext_item_symbol_aux_vii_tl
4049        }
4050      \group_begin:
4051        \tl_use:N \l__enumext_label_font_style_vii_tl
4052        \bool_if:NTF \l__enumext_wrap_label_vii_bool
4053          {
4054            \makebox[ \l__enumext_labelwidth_vii_dim ][ \l__enumext_align_label_vii_str ]
4055              { \__enumext_wrapper_label_vii:n {#1} }
4056          }
4057          {
4058            \makebox[ \l__enumext_labelwidth_vii_dim ][ \l__enumext_align_label_vii_str ]{ #1 }
4059          }
4060      \group_end:
4061      \skip_horizontal:N \l__enumext_labelsep_vii_dim
4062      \tl_use:N \l__enumext_after_list_args_vii_tl
4063      \__enumext_minipage:w [ t ]{ \l__enumext_joined_width_vii_dim  }
4064        \skip_set_eq:NN \parindent \l__enumext_listparindent_vii_dim
4065        \skip_set_eq:NN \parskip \l__enumext_parsep_vii_skip
4066        \tl_use:N \l__enumext_fake_item_indent_vii_tl
4067    }
```

(*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.

```
4068 \cs_new_protected_nopar:Nn \__enumext_stop_item_vii:
4069  {
4070        \__enumext_endminipage:
4071      \endlrbox
4072    \group_end:
4073    \box_set_wd:Nn \l__enumext_item_text_vii_box
4074      {
4075        \l__enumext_joined_width_vii_dim
4076        + \l__enumext_labelwidth_vii_dim
4077        + \l__enumext_labelsep_vii_dim
4078      }
4079    \int_set:Nn \hbadness { 10000 }
4080    \box_use_drop:N \l__enumext_item_text_vii_box
4081    \bool_if:NF \l__enumext_footnotes_key_bool
4082      {
4083        \__enumext_print_footnote:
4084      }
4085    \int_compare:nNnTF { \l__enumext_item_column_pos_vii_int } = { \l__enumext_columns_vii_int }
4086      {
4087        \par\noindent
4088        \int_zero:N \l__enumext_item_column_pos_vii_int
4089      }
4090      { \hspace{ \l__enumext_columns_sep_vii_dim } }
4091    }
```

(*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.

```
4092 \cs_new_protected:Nn \__enumext_remove_extra_parsep_vii:
4093  {
4094    \int_compare:nNnT
4095      {
4096        \int_mod:nn {  \g__enumext_item_count_all_vii_int } { \l__enumext_columns_vii_int }
4097      }
4098      =
4099      { 0 }
4100      {
4101        \par
4102        \vspace{ -\l__enumext_itemsep_vii_skip }
4103        \int_gzero:N \g__enumext_item_count_all_vii_int
4104      }
4105  }
```

(*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.

```
4106 \__enumext_after_env:nn {enumext*} { \__enumext_execute_after_env: }
```

11.40   The environment keyans*

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.

```
4107 \NewDocumentEnvironment{keyans*}{ o }
4108   {
4109     \__enumext_safe_exec_viii:
4110     \__enumext_parse_keys_viii:n {#1}
4111     \__enumext_starred_columns_set_viii:
4112     \__enumext_before_list_viii:
4113     \__enumext_start_list:nn { }
4114       {
4115         \__enumext_list_arg_two_viii:
4116         \__enumext_before_keys_exec_viii:
4117       }
4118     \item[] \scan_stop:
4119     \cs_set_eq:NN \__enumext_stop_item_tmp_viii: \noindent
4120     \cs_set_eq:NN \item \__enumext_start_item_tmp_viii:
4121   }
4122   {
4123     \__enumext_stop_item_tmp_viii:
4124     \__enumext_remove_extra_parsep_viii:
4125     \__enumext_check_starred_cmd:n { item }
4126     \__enumext_stop_list:
4127     \__enumext_after_list_viii:
4128   }
```

(*End of definition for keyans*. This function is documented on page 13.*)

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

```
4129 \cs_new_protected:Nn \__enumext_safe_exec_viii:
4130   {
4131     \int_incr:N \l__enumext_keyans_level_h_int
4132     \int_compare:nNnT { \l__enumext_keyans_level_h_int } > { 1 }
4133       {
4134         \msg_error:nn { enumext } { nested }
4135       }
4136     \__enumext_keyans_start_line:
4137     % Set false for interfering with enumext nested in keyans* (yes, its possible and crayze)
4138     \bool_set_false:N \l__enumext_store_active_bool
4139     \int_compare:nNnT { \l__enumext_level_int } > { 1 }
4140       {
4141         \msg_error:nn { enumext } { keyans-wrong-level }
4142       }
4143   }
```

(*End of definition for* __enumext_safe_exec_viii:.)

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

```
4144 \cs_new_protected:Npn \__enumext_parse_keys_viii:n #1
4145   {
4146     \tl_if_novalue:nF {#1}
4147       {
4148         \keys_set:nn { enumext / keyans* } {#1}
4149       }
4150   }
```

(*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.

```
4151 \cs_new_protected:Nn \__enumext_before_list_viii:
4152   {
4153     \__enumext_vspace_above_viii:
```

```
4154        \__enumext_before_args_exec_viii:
4155        \__enumext_start_mini_viii:
4156      }
```

(*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.

```
4157  \cs_new_protected:Nn \__enumext_after_list_viii:
4158    {
4159      \__enumext_stop_mini_viii:
4160      \__enumext_after_stop_list_viii:
4161      \__enumext_vspace_below_viii:
4162    }
```

(*End of definition for* __enumext_after_list_viii:.)

11.40.1   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.

```
4163  \cs_new_protected_nopar:Nn \__enumext_start_item_tmp_viii:
4164    {
4165      \__enumext_stop_item_tmp_viii:
4166      \int_incr:N \l__enumext_item_column_pos_viii_int
4167      \int_gincr:N \g__enumext_item_count_all_viii_int
4168      \__enumext_item_peek_args_viii:
4169    }
```

(*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).

```
4170  \cs_new_protected:Nn \__enumext_item_peek_args_viii:
4171    {
4172      \peek_meaning:NTF (
4173        { \__enumext_joined_item_viii:w }
4174        { \__enumext_joined_item_viii:w (1) }
4175    }
```

(*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_standar_item_viii:w.

```
4176  \cs_new_protected:Npn \__enumext_joined_item_viii:w (#1)
4177    {
4178      \__enumext_starred_joined_item_viii:n {#1}
4179      \peek_meaning_remove:NTF *
4180        { \__enumext_starred_item_viii:w  }
4181        { \__enumext_standar_item_viii:w }
4182    }
```

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

__enumext_standar_item_viii:w

The function __enumext_standar_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].

```
4183 \cs_new_protected:Npn \__enumext_standar_item_viii:w
4184   {
4185     \bool_set_false:N \l__enumext_item_starred_viii_bool
4186       \peek_meaning:NTF [
4187         {
4188           \bool_set_eq:NN
4189             \l__enumext_wrap_label_viii_bool
4190             \l__enumext_wrap_label_opt_viii_bool
4191           \__enumext_start_item_viii:w
4192         }
4193         {
4194           \bool_set_true:N \l__enumext_wrap_label_viii_bool
4195           \legacy_if_set_true:n { @noitemarg }
4196           \__enumext_start_item_viii:w [ \l__enumext_label_viii_tl ]
4197         }
4198   }
```

(*End of definition for* __enumext_standar_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*⟩].

```
4199 \cs_new_protected:Npn \__enumext_starred_item_viii:w
4200   {
4201     \bool_set_true:N \l__enumext_item_starred_viii_bool
4202     \bool_set_true:N \l__enumext_wrap_label_viii_bool
4203     \peek_meaning:NTF [
4204       { \__enumext_starred_item_viii_aux_i:w }
4205       { \__enumext_starred_item_viii_aux_ii:w }
4206   }
```

The function __enumext_starred_item_viii_aux_i:w will save the optional argument to \item* in \l__enumext_store_current_opt_arg_tl and will save this argument along with the spacing set by the key save-sep in variable \l__enumext_store_current_label_tl if present, then call the function __enumext_starred_item_viii_aux_ii:w.

```
4207 \cs_new_protected:Npn \__enumext_starred_item_viii_aux_i:w [#1]
4208   {
4209     \tl_clear:N \l__enumext_store_current_label_tl
4210     \tl_if_novalue:nF { #1 }
4211       {
4212         \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_tl
4213           {
4214             \tl_put_right:Ne \l__enumext_store_current_label_tl { \l__enumext_store_keyans_item_o|
4215             \tl_put_right:Ne \l__enumext_store_current_label_tl { #1 }
4216           }
4217         \tl_set:Ne \l__enumext_store_current_opt_arg_tl { #1 }
4218       }
4219     \__enumext_starred_item_viii_aux_ii:w
4220   }
4221 \cs_new_protected:Npn \__enumext_starred_item_viii_aux_ii:w
4222   {
4223     \legacy_if_set_true:n { @noitemarg }
4224     \__enumext_start_item_viii:w [ \l__enumext_label_viii_tl ]
4225   }
```

(*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.

```
4226 \cs_new_protected:Nn \__enumext_starred_item_exec:
```

```
4227   {
4228     \tl_put_left:Ne \l__enumext_store_current_label_tl { \l__enumext_label_viii_tl }
4229     \__enumext_store_addto_prop:V \l__enumext_store_current_label_tl
4230     \__enumext_keyans_store_ref:
4231     \tl_put_left:Ne \l__enumext_store_current_label_tl { \item }
4232     \__enumext_keyans_addto_seq_link:
4233     \int_gincr:N \g__enumext_check_starred_cmd_int
4234     \bool_if:NT \l__enumext_show_answer_bool
4235       {
4236         \__enumext_print_keyans_box:NN \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
4237       }
4238     \bool_if:NT \l__enumext_show_position_bool
4239       {
4240         \tl_set:Ne \l__enumext_mark_answer_sym_tl
4241           {
4242             \group_begin:
4243               \exp_not:N \normalfont
4244               \exp_not:N \footnotesize [ \int_eval:n
4245                 {
4246                   \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
4247                 }
4248                 ]
4249             \group_end:
4250           }
4251         \__enumext_print_keyans_box:NN \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
4252       }
4253   }
```

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

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

`\__enumext_start_item_viii:w`   The implementation at this point is very similar to that of the enumext* environment.

```
4254 \cs_new_protected_nopar:Npn \__enumext_start_item_viii:w [#1]
4255   {
4256     \cs_set_eq:NN \__enumext_stop_item_tmp_viii: \__enumext_stop_item_viii:
4257     \legacy_if:nT { @noitemarg }
4258       {
4259         \legacy_if_set_false:n { @noitemarg }
4260         \legacy_if:nT { @nmbrlist }
4261           {
4262             \bool_if:NT \l__enumext_hyperref_bool
4263               {
4264                 \legacy_if_set_true:n { @hyper@item }
4265               }
4266             \refstepcounter{enumXviii}
4267           }
4268       }
```

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

```
4269     \group_begin:
4270       \lrbox{ \l__enumext_item_text_viii_box }
4271         \bool_if:NF \l__enumext_footnotes_key_bool
4272           {
4273             \__enumext_renew_footnote:
4274           }
4275         \bool_if:NT \l__enumext_item_starred_viii_bool
4276           {
4277             \__enumext_starred_item_exec:
4278           }
4279         \group_begin:
4280           \tl_use:N \l__enumext_label_font_style_viii_tl
4281           \bool_if:NTF \l__enumext_wrap_label_viii_bool
4282             {
4283               \makebox[ \l__enumext_labelwidth_viii_dim ][ \l__enumext_align_label_viii_str ]
4284                 { \__enumext_wrapper_label_viii:n {#1} }
4285             }
4286             {
4287               \makebox[ \l__enumext_labelwidth_viii_dim ][ \l__enumext_align_label_viii_str ]{ #1
4288             }
4289         \group_end:
4290         \skip_horizontal:N \l__enumext_labelsep_viii_dim
```

```
4291        \tl_use:N \l__enumext_after_list_args_viii_tl
4292        \__enumext_minipage:w [ t ]{ \l__enumext_joined_width_viii_dim  }
4293          \skip_set_eq:NN \parindent \l__enumext_listparindent_viii_dim
4294          \skip_set_eq:NN \parskip \l__enumext_parsep_viii_skip
4295          \bool_if:NT \l__enumext_item_starred_viii_bool
4296            {
4297              \tl_use:N \l__enumext_fake_item_indent_viii_tl
4298              \__enumext_keyans_show_item_opt:
4299              \skip_horizontal:n { -\l__enumext_fake_item_indent_viii_dim - \l__enumext_labelsep_
4300            }
4301            {
4302              \tl_use:N \l__enumext_fake_item_indent_viii_tl
4303            }
4304      }
```

(*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.

```
4305  \cs_new_protected_nopar:Nn \__enumext_stop_item_viii:
4306    {
4307          \__enumext_endminipage:
4308        \endlrbox
4309      \group_end:
4310      \box_set_wd:Nn \l__enumext_item_text_viii_box
4311        {
4312          \l__enumext_joined_width_viii_dim
4313          + \l__enumext_labelwidth_viii_dim
4314          + \l__enumext_labelsep_viii_dim
4315        }
4316      \int_set:Nn \hbadness { 10000 }
4317      \box_use_drop:N \l__enumext_item_text_viii_box
4318      \bool_if:NF \l__enumext_footnotes_key_bool
4319        {
4320          \__enumext_print_footnote:
4321        }
4322      \int_compare:nNnTF
4323        { \l__enumext_item_column_pos_viii_int } = { \l__enumext_columns_viii_int }
4324        {
4325          \par\noindent
4326          \int_zero:N \l__enumext_item_column_pos_viii_int
4327        }
4328        { \hspace{ \l__enumext_columns_sep_viii_dim } }
4329    }
```

(*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.

```
4330  \cs_new_protected:Nn \__enumext_remove_extra_parsep_viii:
4331    {
4332      \int_compare:nNnT
4333        {
4334          \int_mod:nn
4335            { \g__enumext_item_count_all_viii_int }
4336            { \l__enumext_columns_viii_int }
4337        }
4338        =
4339        { 0 }
4340        {
4341          \par
4342          \vspace{ -\l__enumext_itemsep_viii_skip }
4343          \int_gzero:N \g__enumext_item_count_all_viii_int
4344        }
4345    }
```

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

11.41 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.

```
4346  \NewDocumentCommand \getkeyans { m }
4347    {
4348      \exp_args:Ne \__enumext_getkeyans_aux:n
4349        { \tl_to_str:e { \text_expand:n {#1} } }
4350    }
```

(*End of definition for* \getkeyans. *This function is documented on page 15*.)

__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.

```
4351  \cs_new_protected:Npn \__enumext_getkeyans_aux:n #1
4352    {
4353      \str_if_in:nnTF {#1} { : }
4354        {
4355          \use:e
4356            {
4357              \cs_set:Npn \exp_not:N \__enumext_tmp:w ##1 \c_colon_str ##2 \scan_stop:
4358                { {##1} {##2} }
4359            }
4360          \exp_after:wN \__enumext_getkeyans:nn \__enumext_tmp:w #1 \scan_stop:
4361        }
4362        { \msg_error:nnn { enumext } { missing-colon } {#1} }
4363    }
```

(*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*⟩.

```
4364  \cs_new_protected:Npn \__enumext_getkeyans:nn #1 #2
4365    {
4366      \prop_if_exist:cF { g__enumext_#1_prop }
4367        { \msg_error:nnn { enumext } { undefined-storage-anskey } {#1} }
4368      \group_begin:
4369        \prop_item:cn { g__enumext_#1_prop }{#2}
4370      \group_end:
4371    }
```

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

11.42 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 define a set of ⟨*filtered keys*⟩ with which we will control the options of the different nesting levels for the environment enumext and enumext* by storing their values in the list of tokens \l__enumext_print_keyans_X_tl.

The variable \l__enumext_print_keyans_starred_tl will have the default ⟨*keys*⟩ for \printkeyans* and will be set by \setenumext[⟨*print**⟩] and the variable \l__enumext_print_keyans_vii_tl will have the default keys for the environment enumext* nested within the ⟨*sequence*⟩ and will be set by \setenumext[⟨*print ,**⟩], the rest of the variables will be for the environment enumext and will be set by \setenumext[⟨*print , level*⟩]

```
4372  \cs_generate_variant:Nn \keys_precompile:nnN { neN }
4373  \keys_define:nn  { enumext / print }
4374    {
4375      print*  .code:n     = \keys_precompile:neN { enumext / enumext* }
4376                             { \__enumext_filter_save_key:n {#1} }
4377                             \l__enumext_print_keyans_starred_tl, % starred cmd
4378      print*  .initial:n  = { nosep, label=\arabic*., columns=2, first=\small, font=\small },
4379      print-1 .code:n     = \keys_precompile:neN { enumext / level-1 }
4380                             { \__enumext_filter_save_key:n {#1} }
4381                             \l__enumext_print_keyans_i_tl,
4382      print-1 .initial:n  = { nosep, label=\arabic*., columns=2, first=\small, font=\small },
4383      print-2 .code:n     = \keys_precompile:neN { enumext / level-2 }
4384                             { \__enumext_filter_save_key:n {#1} }
4385                             \l__enumext_print_keyans_ii_tl,
4386      print-2 .initial:n  = { nosep, label=(\alph*), first=\small, font=\small },
4387      print-3 .code:n     = \keys_precompile:neN { enumext / level-3 }
```

```
4388                                  { \__enumext_filter_save_key:n {#1} }
4389                                  \l__enumext_print_keyans_iii_tl,
4390        print-3 .initial:n  = { nosep, label=\roman*., first=\small, font=\small },
4391        print-4 .code:n     = \keys_precompile:neN { enumext / level-4 }
4392                                  { \__enumext_filter_save_key:n {#1} }
4393                                  \l__enumext_print_keyans_iv_tl,
4394        print-4 .initial:n  = { nosep, label=\Alph*., first=\small, font=\small },
4395        print-* .code:n     = \keys_precompile:neN { enumext / enumext* }
4396                                  { \__enumext_filter_save_key:n {#1} }
4397                                  \l__enumext_print_keyans_vii_tl, % starred nested
4398        print-* .initial:n  = { nosep, label=\arabic*., first=\small, font=\small },
4399      }
```

💣 The reason for storing ⟨keys⟩ in token lists using \keys_precompile:neN is because the keys are set via \setenumext but are later executed by running the command \printkeyans and they are not handled directly by its optional argument, except those related to the first opening level.

\printkeyans Create a user command to print *"all stored content"* in ⟨sequence⟩ for \anskey, anskey*, \item* and \anspic*. Within a group we will run our *"precompiled keys"* and then call the internal function __enumext_printkeyans:nnn.

```
4400  \NewDocumentCommand \printkeyans { s O{} m }
4401    {
4402      \group_begin:
4403        \tl_use:N \l__enumext_print_keyans_i_tl
4404        \tl_use:N \l__enumext_print_keyans_ii_tl
4405        \tl_use:N \l__enumext_print_keyans_iii_tl
4406        \tl_use:N \l__enumext_print_keyans_iv_tl
4407        \tl_use:N \l__enumext_print_keyans_vii_tl
4408        \__enumext_printkeyans:nnn { #1 } { #2 } { #3 }
4409      \group_end:
4410    }
```

(*End of definition for* \printkeyans. *This function is documented on page 16.*)

__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 check if not empty.

```
4411  \cs_new_protected:Npn \__enumext_printkeyans:nnn #1 #2 #3
4412    {
4413      \seq_if_exist:cTF { g__enumext_#3_seq }
4414        {
4415          \seq_if_empty:cF { g__enumext_#3_seq }
4416            {
4417              %%\seq_show:c { g__enumext_#3_seq }
```

If the starred argument is present we will check that the environment enumext* is not saved in the ⟨sequence⟩, then execute the variable \l__enumext_print_keyans_starred_tl that contains the default ⟨keys⟩ for the environment enumext*, it will open the environment enumext* passing the optional argument to the *"first level"*, set the key base-fix and then will map the ⟨sequence⟩.

```
4418              \bool_if:nTF {#1}
4419                {
4420                  \seq_if_in:cnTF { g__enumext_#3_seq } { \end{enumext*} }
4421                    {
4422                      \msg_error:nnnn { enumext } { print-starred } {#3} { enumext* }
4423                    }
4424                    {
4425                      \tl_use:N \l__enumext_print_keyans_starred_tl
4426                      \begin{enumext*}[#2]
4427                        \keys_set:nn { enumext / level-1 }{ base-fix }
4428                        \seq_map_inline:cn { g__enumext_#3_seq } { ##1 }
4429                      \end{enumext*}
4430                    }
4431                }
```

Otherwise it will open the environment enumext passing the optional argument to the *"first level"*, set the key base-fix and then map the ⟨sequence⟩.

```
4432                {
4433                  \begin{enumext}[#2]
4434                    \keys_set:nn { enumext / enumext* }{ base-fix }
4435                    \seq_map_inline:cn { g__enumext_#3_seq } { ##1 }
4436                  \end{enumext}
4437                }
```

```
4438                      }
4439                  }
4440                  {
4441                      \msg_error:nnn { enumext } { undefined-storage-anskey } {#3}
4442                  }
4443          }
```

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

11.43   The command \setenumext

The command \setenumext will be in charge of managing the ⟨*keys*⟩ passed to all environments and to the \printkeyans command. We must take precautions with the enumext* environment and *"first level"* of the enumext environment so as not to capture ⟨*keys*⟩ that complicate us.

__enumext_filter_first_level:n
__enumext_filter_first_level_key:n
__enumext_filter_first_level_pair:nn

The function __enumext_filter_first_level:n will be in charge of filtering the ⟨*keys*⟩ passed to the environment enumext* and *"first level"* of the environment enumext.

```
4444  \cs_new:Npn \__enumext_filter_first_level:n #1
4445    {
4446      \use:e
4447        {
4448          \keyval_parse:NNn
4449            \__enumext_filter_first_level_key:n
4450            \__enumext_filter_first_level_pair:nn {#1}
4451        }
4452    }
```

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

```
4453  \cs_new:Npn \__enumext_filter_first_level_key:n #1
4454    {
4455      \str_case:nnF {#1}
4456        {
4457          { resume   } {}
4458          { resume*  } {}
4459        }
4460        { , { \exp_not:n {#1} } }
4461    }
```

The function __enumext_filter_first_level_pair:nn will be responsible for filtering the ⟨*keys*⟩ that are passed *"with value"* by excluding the series, resume and save-ans keys.

```
4462  \cs_new:Npn \__enumext_filter_first_level_pair:nn #1#2
4463    {
4464      \str_case:nnF {#1}
4465        {
4466          { series } {}
4467          { resume } {}
4468          { save-ans } {}
4469        }
4470        { , { \exp_not:n {#1} } = { \exp_not:n {#2} } }
4471    }
```

(*End of definition for __enumext_filter_first_level:n, __enumext_filter_first_level_key:n, and __enumext_-filter_first_level_pair:nn.*)

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

```
4472  \keys_define:nn { enumext / meta-families }
4473    {
4474      enumext-1 .code:n =
4475                    {
4476                        \keys_set:ne { enumext / level-1  }
4477                          {
4478                              \__enumext_filter_first_level:n {#1}
4479                          }
4480                    } ,
4481      enumext-2 .code:n = { \keys_set:nn { enumext / level-2  } {#1} } ,
4482      enumext-3 .code:n = { \keys_set:nn { enumext / level-3  } {#1} } ,
4483      enumext-4 .code:n = { \keys_set:nn { enumext / level-4  } {#1} } ,
4484      keyans    .code:n = { \keys_set:nn { enumext / keyans   } {#1} } ,
4485      enumext*  .code:n =
4486                    {
4487                        \keys_set:ne { enumext / enumext* }
4488                          {
```

```
4489                      \__enumext_filter_first_level:n {#1}
4490                    }
4491                  } ,
4492        keyans*    .code:n = { \keys_set:nn { enumext / keyans*  } {#1} } ,
4493        print*     .code:n = { \keys_set:nn { enumext / print    } { print*  = {#1} } } ,
4494        print-1    .code:n = { \keys_set:nn { enumext / print    } { print-1 = {#1} } } ,
4495        print-2    .code:n = { \keys_set:nn { enumext / print    } { print-2 = {#1} } } ,
4496        print-3    .code:n = { \keys_set:nn { enumext / print    } { print-3 = {#1} } } ,
4497        print-4    .code:n = { \keys_set:nn { enumext / print    } { print-4 = {#1} } } ,
4498        print-*    .code:n = { \keys_set:nn { enumext / print    } { print-* = {#1} } } ,
4499        unknown    .code:n = { \msg_error:nn { enumext } { unknown-key-family } } ,
4500      }
```

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

```
4501  \seq_const_from_clist:Nn \c__enumext_all_families_seq
4502    {
4503      enumext-1, enumext-2, enumext-3, enumext-4, keyans, enumext*,
4504      keyans*, print-1, print-2, print-3, print-4, print-*, print*,
4505    }
```

\setenumext Now we define the user command \setenumext.

```
4506  \NewDocumentCommand \setenumext { O{enumext,1} +m }
4507    {
4508      \tl_if_novalue:nTF {#1}
4509        {
4510          \seq_map_inline:Nn \c__enumext_all_families_seq
4511        }
4512        {
4513          \seq_clear:N \l__enumext_setkey_tmpa_seq
4514          \seq_set_from_clist:Nn \l__enumext_setkey_tmpb_seq {#1}
4515          \int_set:Nn \l__enumext_setkey_tmpa_int
4516            {
4517              \seq_count:N \l__enumext_setkey_tmpb_seq
4518            }
4519          \int_compare:nNnTF { \l__enumext_setkey_tmpa_int } > { 1 }
4520            {
4521              \seq_pop_left:NN \l__enumext_setkey_tmpb_seq \l__enumext_setkey_tmpa_tl
4522              \seq_map_function:NN \l__enumext_setkey_tmpb_seq \__enumext_set_parse:n
4523              \seq_set_map_e:NNn \l__enumext_setkey_tmpa_seq \l__enumext_setkey_tmpa_seq
4524                {
4525                  \tl_use:N \l__enumext_setkey_tmpa_tl - ##1
4526                }
4527            }
4528            {
4529              \seq_put_right:Ne \l__enumext_setkey_tmpa_seq { \tl_trim_spaces:n {#1} }
4530            }
4531          \seq_if_empty:NTF \l__enumext_setkey_tmpa_seq
4532            { \seq_map_inline:Nn \c__enumext_all_families_seq }
4533            { \seq_map_inline:Nn \l__enumext_setkey_tmpa_seq }
4534        }
4535        {
4536          \keys_set:nn { enumext / meta-families } { ##1 = {#2} }
4537        }
4538    }
```

(*End of definition for* \setenumext*. This function is documented on page 6.*)

__enumext_set_parse:n
__enumext_set_error:nn

Internal functions used by the \setenumext command.

```
4539  \cs_new_protected:Npn \__enumext_set_parse:n #1
4540    {
4541      \tl_set:Ne \l__enumext_setkey_tmpb_tl { \tl_trim_spaces:n {#1} }
4542      \clist_map_inline:nn { 0, 1, 2, 3, 4, * } % <- max level
4543        { \tl_remove_all:Nn \l__enumext_setkey_tmpb_tl {##1} }
4544      \tl_if_empty:NTF \l__enumext_setkey_tmpb_tl
4545        {
4546          \seq_put_right:Ne \l__enumext_setkey_tmpa_seq
4547            { \tl_trim_spaces:n {#1} }
4548        }
4549        { \__enumext_set_error:nn {#1} { } }
4550    }
4551  \cs_new_protected:Npn \__enumext_set_error:nn #1 #2
4552    { \msg_error:nnn { enumext } { invalid-key } {#1} {#2} }
```

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

11.44   Messages

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

```
4553 \msg_new:nnn { enumext } { package-load }
4554   {
4555     The ~ '#1' ~ package ~ is ~ already ~ loaded.
4556   }
4557 \msg_new:nnn { enumext } { package-not-load }
4558   {
4559     The ~ '#1' ~ package ~ will ~ be ~ loaded ~ as ~ a ~ dependency.
4560   }
4561 \msg_new:nnn { enumext } { package-load-foot }
4562   {
4563     The ~ '#1' ~ package ~ is ~ loaded ~ with ~ the ~ option ~ '#2'.
4564   }
```

Message used in the creation of counters by enumext package.

```
4565 \msg_new:nnn { enumext } { counters }
4566   {
4567     The ~ counter ~ '#1' ~ is ~ already ~ defined ~ by ~ some ~ \\
4568     package ~ or ~ macro, ~ it ~ cannot ~ be ~ continued.
4569   }
```

Message used by align and mark-pos keys.

```
4570 \msg_new:nnn { enumext } { unknown-choice }
4571   {
4572     The ~ value ~ '#3' ~ for ~ '#1' ~ key ~ is ~ invalid ~ use ~ ('#2').
4573   }
```

Message used by reserved anskey* environment by enumext package.

```
4574 \msg_new:nnnn { enumext } { anskey-env-error }
4575   {
4576     The ~ '#1' ~ environment ~is ~ reserved ~ by ~\\
4577     'enumext' ~ package, ~ It~ is~ already~ defined.
4578   }
4579   {
4580     The ~ anskey* ~ environment ~ is ~ defined ~ internally ~
4581     for ~ the ~ 'save-ans' ~ key.\\
4582   }
```

Message used in the creation of ⟨*prop list*⟩ by enumext package.

```
4583 \msg_new:nnn { enumext } { store-prop }
4584   {
4585     * ~ Package ~ enumext: ~ Creating ~
4586     \c_backslash_str g__enumext_#1_prop ~ \msg_line_context:.
4587   }
4588 \msg_new:nnn { enumext } { store-seq }
4589   {
4590     * ~ Package ~ enumext: ~ Creating ~
4591     \c_backslash_str g__enumext_#1_seq ~ \msg_line_context:.
4592   }
4593 \msg_new:nnn { enumext } { store-int }
4594   {
4595     * ~ Package ~ enumext: ~ Creating ~
4596     \c_backslash_str g__enumext_resume_#1_int ~ \msg_line_context:.
4597   }
4598 \msg_new:nnn { enumext } { prop-seq-int-hook }
4599   {
4600     * ~ Package ~ enumext: ~ Elements ~ in ~
4601     \c_backslash_str g__enumext_#1_prop ~ = ~ #2.\\
4602     * ~ Package ~ enumext: ~ Elements ~ in ~
4603     \c_backslash_str g__enumext_#1_seq ~ = ~ #3.\\
4604     * ~ Package ~ enumext: ~ Value ~ off ~
4605     \c_backslash_str g__enumext_resume_#1_int ~ = ~ #4.
4606   }
4607 \msg_new:nnn { enumext } { item-answer-hook }
4608   {
4609     * ~ Package ~ enumext: ~ Value ~ off ~
4610     \c_backslash_str g__enumext_item_number_int ~ = ~ #1.\\
4611     * ~ Package ~ enumext: ~ Value ~ off ~
4612     \c_backslash_str g__enumext_item_anskey_int ~ = ~ #2.\\
```

```
4613        * ~ Package ~ enumext: ~ Difference ~ item_number_int ~ - ~ item_anskey_int ~ = ~ #3.
4614      }
```

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

```
4615  \msg_new:nnn { enumext } { invalid-key }
4616    {
4617      The ~ key ~ '#1' ~ is ~ not ~ know ~ the ~ level ~ #2.
4618    }
4619  \msg_new:nnn { enumext } { unknown-key-family }
4620    {
4621      Unknown~key~family~`\l_keys_key_str'~for~enumext.
4622    }
```

Messages used in length calculation.

```
4623  \msg_new:nnn { enumext } { width-negative }
4624    {
4625      Ignoring ~ negative ~ value ~ '#1=#2' ~ \msg_line_context:.\\
4626      The ~ key ~ '#1'~ accepts ~ values  ~ >= ~ 0pt.
4627    }
4628  \msg_new:nnn { enumext } { width-zero }
4629    {
4630      Invalid ~ '#1=#2' ~ \msg_line_context:.\\
4631      The ~ key ~ '#1'~ accepts ~ values  ~ > ~ 0pt.
4632    }
```

Messages used by show-length key in enumext.

```
4633  \msg_new:nnn { enumext } { list-lengths }
4634    {
4635      **** ~ Lengths ~ used ~ by ~ 'enumext' ~ level ~ '#2' ~ \msg_line_context:~\c_space_tl ****\\
4636      \__enumext_show_length:nnn { dim  } { labelsep     } {#1}
4637      \__enumext_show_length:nnn { dim  } { labelwidth   } {#1}
4638      \__enumext_show_length:nnn { dim  } { itemindent   } {#1}
4639      \__enumext_show_length:nnn { dim  } { leftmargin   } {#1}
4640      \__enumext_show_length:nnn { dim  } { rightmargin  } {#1}
4641      \__enumext_show_length:nnn { dim  } { listparindent } {#1}
4642      \__enumext_show_length:nnn { skip } { topsep     } {#1}
4643      \__enumext_show_length:nnn { skip } { parsep     } {#1}
4644      \__enumext_show_length:nnn { skip } { partopsep } {#1}
4645      \__enumext_show_length:nnn { skip } { itemsep    } {#1}
4646      ***************************************************
4647    }
```

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

```
4648  \msg_new:nnn { enumext } { list-lengths-not-nested }
4649    {
4650      **** ~ Lengths ~ used ~ by ~ '#2' ~ environment ~ \msg_line_context:~\c_space_tl ****\\
4651      \__enumext_show_length:nnn { dim  } { labelsep     } {#1}
4652      \__enumext_show_length:nnn { dim  } { labelwidth   } {#1}
4653      \__enumext_show_length:nnn { dim  } { itemindent   } {#1}
4654      \__enumext_show_length:nnn { dim  } { leftmargin   } {#1}
4655      \__enumext_show_length:nnn { dim  } { rightmargin  } {#1}
4656      \__enumext_show_length:nnn { dim  } { listparindent } {#1}
4657      \__enumext_show_length:nnn { skip } { topsep     } {#1}
4658      \__enumext_show_length:nnn { skip } { parsep     } {#1}
4659      \__enumext_show_length:nnn { skip } { partopsep } {#1}
4660      \__enumext_show_length:nnn { skip } { itemsep    } {#1}
4661      ***************************************************
4662    }
```

Messages used by ref key.

```
4663  \msg_new:nnn { enumext } { key-ref-empty }
4664    {
4665      Key ~ 'ref' ~ need ~ a ~ value ~ in ~ '#1'~ \msg_line_context:.
4666    }
```

Messages used by save-ans key.

```
4667  \msg_new:nnn { enumext } { save-ans-empty }
4668    {
4669      Key ~ 'save-ans' ~ need ~ a ~ value ~ in ~ '#1'~ \msg_line_context:.
4670    }
4671  \msg_new:nnn { enumext } { save-ans-log }
4672    {
4673      * ~ Package ~ enumext: ~ Start ~ #1\c_space_tl with ~ save-ans=#2 ~ \msg_line_context:.
4674    }
```

```
4675  \msg_new:nnn { enumext } { save-ans-log-hook }
4676    {
4677      * ~ Package ~ enumext: ~ Stop ~ #1\c_space_tl with ~ save-ans=#2 ~ \msg_line_context:.
4678    }
4679  \msg_new:nnn { enumext } { save-ans-hook }
4680    {
4681      Stop ~ storing ~ for ~ 'save-ans=#1' ~ \msg_line_context:.
4682    }
```

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

```
4683  \msg_new:nnn { enumext } { need-save-ans }
4684    {
4685      Key ~ '#1'~ works ~ only ~ with ~ the ~ 'save-ans' ~ key ~ in ~ '#2'~ \msg_line_context:.
4686    }
4687  \msg_new:nnn { enumext } { items-same-answer }
4688    {
4689      ****************************************\\
4690      * ~ Package ~ enumext: ~ Checking ~ answers ~ in ~ '#1' ~
4691      for ~ \c_left_brace_str #2 \c_right_brace_str\\
4692      * ~ started ~ #3 ~ and ~ close ~ \msg_line_context: : ~
4693      'OK', ~ all ~ items ~ with ~ answer.\\
4694      ****************************************
4695    }
4696  \msg_new:nnn { enumext } { item-greater-answer }
4697    {
4698      Checking ~ answers ~ in ~ '#1' ~ for ~ \c_left_brace_str #2 \c_right_brace_str\\
4699      started ~ #3 ~ and ~ close ~ \msg_line_context: : ~'NOT ~ OK'\\
4700      Items ~ > ~ Answers.
4701    }
4702  \msg_new:nnn { enumext } { item-less-answer }
4703    {
4704      Checking ~ answers ~ in ~ '#1' ~ for ~ \c_left_brace_str #2 \c_right_brace_str\\
4705      started ~ #3 ~ and ~ close ~ \msg_line_context: : ~'NOT ~ OK'\\
4706      Items ~ < ~ Answers.
4707    }
```

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

```
4708  \msg_new:nnn { enumext } { missing-starred }
4709    {
4710      Missing ~ '\c_backslash_str #1*' ~ #2.
4711    }
4712  \msg_new:nnn { enumext } { many-starred }
4713    {
4714      Many ~ '\c_backslash_str #1*' ~ #2.
4715    }
```

Messages used by \printkeyans* command.

```
4716  \msg_new:nnn { enumext } { print-starred }
4717    {
4718      \c_backslash_str printkeyans*:~ The ~ sequence ~ '#1' ~ already ~ contains ~
4719      #2 ~ environment ~  \msg_line_context:.
4720    }
```

Message for the nesting depth of the environment enumext.

```
4721  \msg_new:nnn { enumext } { list-too-deep }
4722    {
4723      Too ~ deep ~ nesting  ~ for ~ 'enumext' ~ \msg_line_context:.~ \\
4724      The ~ maximum  ~ level  ~ of  ~ nesting  ~ is ~ 4.
4725    }
```

Messages used by \anskey, anskey* and \anspic commands.

```
4726  \msg_new:nnn { enumext } { anskey-unnumber-item }
4727    {
4728      Can't ~ store ~ with ~ a ~ unnumbered ~ \c_backslash_str item ~ \msg_line_context:.
4729    }
4730  \msg_new:nnn { enumext } { anskey-already-stored }
4731    {
4732      Content ~ already ~ stored ~ for ~ this ~ \c_backslash_str item ~ \msg_line_context:.
4733    }
4734  \msg_new:nnn { enumext } { anskey-empty-arg }
4735    {
4736      Can't ~ store ~ empty ~ content ~ ~ \msg_line_context:.
4737    }
```

```
4738  \msg_new:nnn { enumext } { anskey-wrong-place }
4739    {
4740      Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:.~ \\
4741      '\c_backslash_str #1' ~ works ~ in ~ the ~ environment ~ '#2'.
4742    }
4743  \msg_new:nnn { enumext } { anskey-nested }
4744    {
4745      The ~ command ~ \c_backslash_str anskey~ can't ~ be ~ nested ~ \msg_line_context:.
4746    }
4747  \msg_new:nnn { enumext } { anskey-math-mode }
4748    {
4749      #1 ~ can't ~ work ~ in ~ math ~ mode ~ \msg_line_context:.
4750    }
4751  \msg_new:nnn { enumext } { anskey-env-wrong }
4752    {
4753      The ~ environment ~ anskey* ~ cannot ~ use ~ in ~ '#1' ~ \msg_line_context:.
4754    }
4755  \msg_new:nnn { enumext } { anspic-wrong-place }
4756    {
4757      Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:.~ \\
4758      '\c_backslash_str #1' ~ works ~ in ~ the ~ environment ~ '#2'.
4759    }
4760  \msg_new:nnn { enumext } { command-wrong-place }
4761    {
4762      Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:.~ \\
4763      '\c_backslash_str #1' ~ works ~ outside ~ the ~ environment ~ '#2'.
4764    }
4765  \msg_new:nnnn { enumext } { anskey-env-key-unknown }
4766    {
4767      The ~ key ~ '#1' ~ is ~ unknown ~ by ~ environment~
4768      'anskey*' ~ and ~ is ~ being ~ ignored.
4769    }
4770    {
4771      The ~ environment ~ 'anskey*' ~ does ~ not ~ have ~ a ~ key ~ called ~'#1'.\\
4772      Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
4773    }
4774  \msg_new:nnnn { enumext } { anskey-env-key-value-unknown }
4775    {
4776      The ~ key ~ '#1=#2' ~ is ~ unknown ~ by ~ environment ~
4777      'anskey*' ~ and ~ is ~ being ~ ignored.
4778    }
4779    {
4780      The ~ environment ~ 'anskey*' ~ does ~ not ~ have ~ a ~ key ~ called ~'#1'.\\
4781      Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
4782    }
4783  \msg_new:nnnn { enumext } { anskey-cmd-key-unknown }
4784    { The~key~'#1'~is~unknown~by~'\c_backslash_str anskey'~and~is~being~ignored.}
4785    {
4786      The~command~'\c_backslash_str anskey'~does~not~have~a~key~called~'#1'.\\
4787      Check~that~you~have~spelled~the~key~name~correctly.
4788    }
4789  \msg_new:nnnn { enumext } { anskey-cmd-key-value-unknown }
4790    { The ~ key ~ '#1=#2' ~ is ~ unknown ~ by ~ '\c_backslash_str anskey' ~ and ~ is ~ being ~ ignor
4791    {
4792      The ~ command ~ '\c_backslash_str anskey' ~ does ~ not ~ have ~ a ~ key ~ called ~'#1'.\\
4793      Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
4794    }
```

Messages used by keyans and keyanspic environment.

```
4795  \msg_new:nnn { enumext } { keyans-nested }
4796    {
4797      The ~ environment ~ 'keyans' ~ can't ~ be  ~ nested  ~ \msg_line_context:.
4798    }
4799  \msg_new:nnn { enumext } { keyans-wrong-level }
4800    {
4801      Wrong ~ level ~ position ~ for ~ 'keyans' ~ \msg_line_context:.~ \\
4802      The ~ environment ~ 'keyans' ~ can ~ only ~ be ~ in ~ the ~ first ~ level.
4803    }
4804  \msg_new:nnn { enumext } { wrong-place }
4805    {
4806      Wrong ~ place ~ for ~ '#1' ~ environment ~\msg_line_context:.~ \\
4807      '#1' ~ is ~ only ~ found ~ with ~ '#2' ~  in  ~  'enumext.
```

```
4808        }
4809  \msg_new:nnn { enumext } { keyanspic-nested }
4810        {
4811          The ~ environment ~ 'keyanspic' ~ can't ~ be  ~ nested~ \msg_line_context:.~.
4812        }
4813  \msg_new:nnn { enumext } { keyanspic-wrong-level }
4814        {
4815          Wrong ~ level ~ position ~ for ~ 'keyanspic' ~ \msg_line_context:.~ \\
4816          The ~ environment ~ 'keyans' ~ can ~ only ~ be ~ in ~ the ~ first ~ level.
4817        }
```

Messages used by \getkeyans command.

```
4818  \msg_new:nnn { enumext } { undefined-storage-anskey }
4819        {
4820          Storage ~ named ~ '#1' ~ is ~ not ~ defined ~ \msg_line_context:.
4821        }
```

Messages used by \miniright command.

```
4822  \msg_new:nnn { enumext } { missing-miniright }
4823        {
4824          Missing ~ '\c_backslash_str miniright' ~ in ~ \msg_line_context:.\\
4825          The ~ key ~ 'mini-env' ~ need ~ '\c_backslash_str miniright'.
4826        }
4827  \msg_new:nnn { enumext } { wrong-miniright-place }
4828        {
4829          Wrong ~ place ~ for ~ '\c_backslash_str miniright' ~ \msg_line_context:.~ \\
4830          Works ~ in ~ 'enumext' ~ and ~ 'keyans' ~ with ~ key ~ 'mini-env'.
4831        }
4832  \msg_new:nnn { enumext } { wrong-miniright-use }
4833        {
4834          Wrong ~ use ~ for ~ '\c_backslash_str miniright' ~ \msg_line_context:.~ \\
4835          '\c_backslash_str miniright' ~ need ~ a ~ key ~ 'mini-env'.
4836        }
```

Messages used by enumext* and keyans* environments.

```
4837  \msg_new:nnn { enumext } { nested }
4838        {
4839          The ~ starred ~ environment ~ can't ~ be ~ nested ~ \msg_line_context:.
4840        }
4841  \msg_new:nnn { enumext } { item-joined }
4842        {
4843          Items ~ joined ~ (#1) ~ > ~ #2  ~ columns ~\msg_line_context:.
4844        }
4845  \msg_new:nnn { enumext } { item-joined-columns }
4846        {
4847          Not ~ space ~ to ~ join ~ items ~ (#1) ~ > ~ #2 ~\msg_line_context:.
4848        }
```

11.45   Finish package

Finish package implementation.

```
4849  \file_input_stop:
4850  ⟨/package⟩
```

12 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.