enumext

ENUMERATE EXERCISE SHEETS

V1.0   2024-06-07[*]

©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 2024-06-07.
[†]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 in `enumext`, `enumext*`, `anskeyenv`, `keyans`, `keyans*` and `keyanspic` environments, `\anskey`, `\item*` and `\anspic*` commands to *stored content*, `\getkeyans` command to get the individual *stored content*, `\printkeyans` to print all *stored content*, `\miniright` for `minipage` and `\setenumext` to config all [⟨*key = val*⟩] options.

1.3.1 Internal counters

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

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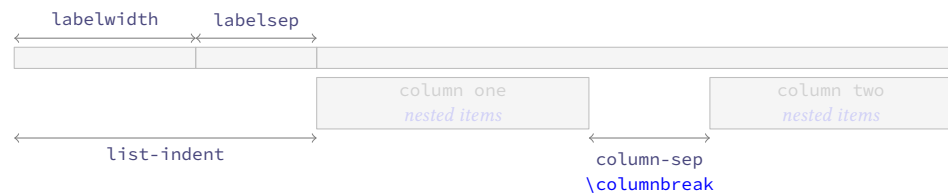

Figure 4: Representation of the two column output for a nested level in `enumext` environment.

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. The environment is available for all nesting levels, and can can together with the `mini-env` key. If you need to force a start a new column `\columnbreak` must be used (see §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 (left and right) is always used with *"aligned on top"* [t], the `minipage` environment on the *"right side"* always starts with `\centering`. It can be used at all nesting levels and is controlled by `mini-env` and `mini-sep` keys. In order to switch from the *"left"* side `minipage` environment to the *"right"* side one must use the command `\miniright` (see §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⟩]                    \begin{enumext*}[⟨keyval list⟩]
  \item  ⟨item content⟩                             \item  ⟨item content⟩
  \item [⟨custom⟩]  ⟨item content⟩                  \item [⟨custom⟩]  ⟨item content⟩
  \item*[⟨symbol⟩][⟨offset⟩]  ⟨item content⟩        \item*[⟨symbol⟩][⟨offset⟩]  ⟨item content⟩
\end{enumext}                                     \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* environment 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. 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 nineth
  \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 two columns   Z The nineth

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.

💣 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`
`\miniright*`

The `\miniright` command close the `minipage` environment on the "*left side*" and opens the `minipage` environment on the "*right side*" by starting it with the `\centering` command. It must be placed "*after*" the last `\item` of the current environment and "*before*" starting the material to be placed on the "*right side*". The *starred 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.

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

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

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

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

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 `anskeyenv`, `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 anskeyenv 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 anskeyenv 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 *secuence* 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 *secuence* 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 anskeyenv 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, anskeyenv, \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`, `anskeyenv`, `\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 `anskeyenv`. 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 `anskeyenv`, *"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 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 `\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 anskeyenv

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

The environment anskeyenv 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 `\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 anskeyenv environment is implemented using the `scontents` package, for the correct operation `\begin{anskeyenv}` and `\end{anskeyenv}` 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{anskeyenv}` 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{anskeyenv}` on the same line the package `scontents` will return a "warning" message.

The anskeyenv 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{anskeyenv}[item-star]
      ⟨first answer⟩
    \end{anskeyenv}
  \item Text containing our instructions or questions.
    \begin{enumext}
      \item Question.
        \begin{anskeyenv}
          ⟨second answer⟩
        \end{anskeyenv}
    \end{enumext}
  \item Text containing our instructions or questions.
    \begin{anskeyenv}
      ⟨third answer⟩
    \end{anskeyenv}
  \item Text containing our instructions or questions.
    \begin{anskeyenv}
      ⟨fourth answer⟩
    \end{anskeyenv}
\end{enumext}
```

★ 5. Text containing our instructions or questions.
   [5] First answer with `verbatim`
6. Text containing our instructions or questions.
   (a) Question.
      [6] second answer

7. Text containing our instructions or questions.
   [7] third answer
8. Text containing our instructions or questions.
   [8] fourth answer

5.4 The environments keyans and keyans*

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

The keyans 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
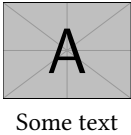
5.5 The environment `keyanspic`

`keyanspic`

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

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

The optional argument determines the number drawings or tabular *"above"* and *"below"* within the environment. The vertical separation between *"above"* and *"below"* is controlled by the values set by

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

`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 ⟨*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.
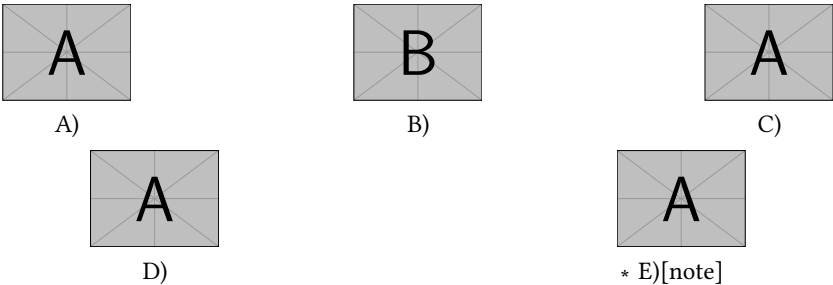
5.6 Printing stored content

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

```latex
\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.
    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   *

---

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

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

Example 3

A *"simple multiple choice"* test 📄.

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

(A)

(B)

(C)

(D)

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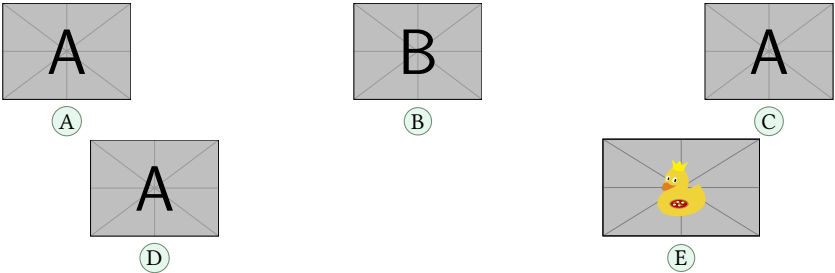
5. Question with image on left side:
   - (A) value
   - (B) value
   - (C) value
   - (D) correct
   - (E) value

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

Example 4

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

🦆1   Factor $x^2 - 2x + 1$

🦆2   Factor $3x + 3y + 3z$

The following questions need to be cuaqtified :)

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

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

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

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

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

Example 5

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

<table>
<tr><td>1</td><td>3</td></tr>
</table>

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=0`pt you get *widest style*:

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

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

Description indented by label

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

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

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

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

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

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

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

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

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

Description with multi-line labels

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

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

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

SomeThing A short one-line description.

This is an entry *without* a label.

Something A short one-line description.

Something long A much longer description. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris.

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris.

SoMeThInG LoNg A much longer description. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris.

Final notes

The original implementation (if you can call it that) of the ideas that led to the creation of enumext were some macros using the enumerate[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 typesetting a document with LaTeX, that is to say the output does not always look as nice as it should, even if they are only alternatives these must follow a certain order when presented either numerical or presentation, that said handling that using *nested lists* is quite complicated so I do not classify to be implemented.

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 2_ε: 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 xparse package". Available from CTAN, https://www.ctan.org/pkg/xparse, 2024.

[13] GUNDLACH, PATRICK. "The lua-visual-debug package". Available from CTAN, https://www.ctan.org/pkg/lua-visual-debug, 2023.

[14] LEMVIG, MOGENS. "The shortlst package". Available from CTAN, https://www.ctan.org/pkg/shortlst, 1998.

[15] NIEDERBERGER, CLEMENS. "tasks – Horizontally columned lists". Available from CTAN, https://www.ctan.org/pkg/tasks, 2022.

9   Change history

v1.0   2024-06-07   – 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    {2024-06-07}
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_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_keyans_env_bool
41 \tl_new:N   \g__enumext_start_line_tl
42 \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).

```
43 \cs_set_protected:Npn \__enumext_tmp:n #1
44   {
45     \tl_new:c { l__enumext_counter_#1_tl }
46   }
47 \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).

```
48 \tl_const:Nn \c__enumext_counter_style_tl
49   { { arabic } { roman } { Roman } { alph } { Alph } }
50 \tl_new:N \l__enumext_ref_key_arg_tl
51 \tl_new:N \l__enumext_ref_the_count_tl
52 \cs_set_protected:Npn \__enumext_tmp:n #1
53   {
54     \tl_new:c  { l__enumext_renew_the_count_#1_tl }
55     \tl_new:c  { l__enumext_the_counter_#1_tl }
56     \tl_set:ce { l__enumext_the_counter_#1_tl } { \exp_not:c { theenumX#1 } }
57   }
58 \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 (§11.22).

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

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

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.

```
65  \dim_new:N \l__enumext_current_widest_dim
66  \tl_new:N  \g__enumext_counter_styles_tl
67  \tl_new:N  \g__enumext_widest_label_tl
68  \box_new:N \l__enumext_label_width_by_box
```

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

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 (§11.15). The variables `\l__-enumext_leftmargin_X_dim` and `\l__enumext_itemindent_X_dim` are used and set by the function `\__enumext_calc_hspace:NNNNNNNNNNNNN` (§11.33.1).

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

Internal variables used by columns key §11.19).

```
77  \cs_set_protected:Npn \__enumext_tmp:n #1
78    {
79      \skip_new:c  { l__enumext_multicols_above_#1_skip }
80      \skip_new:c  { l__enumext_multicols_below_#1_skip }
81    }
82  \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`.*)*

Internal variables used by `\miniright` command (§11.20.4) and the keys mini-right, mini-right*, mini-env and mini-sep (§11.18, §11.20).

```
83  \int_new:N  \g__enumext_minipage_stat_int
84  \skip_new:N \l__enumext_minipage_left_skip
85  \skip_new:N \l__enumext_minipage_right_skip
86  \skip_new:N \l__enumext_minipage_after_skip
87  \skip_new:N \g__enumext_minipage_right_skip
88  \skip_new:N \g__enumext_minipage_after_skip
89  \cs_set_protected:Npn \__enumext_tmp:n #1
90    {
91      \dim_new:c  { l__enumext_minipage_left_#1_dim     }
92      \bool_new:c { l__enumext_minipage_active_#1_bool }
93    }
94  \clist_map_inline:nn { i, ii, iii, iv, v, vii, viii } { \__enumext_tmp:n {#1} }
```

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

The 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 (§11.15.1), 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 (§11.17).

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

(End of definition for \l__enumext_wrap_label_X_bool and others.)

`\l__enumext_store_active_bool`
`\l__enumext_store_name_tl`
`\g__enumext_store_name_tl`
`\l__enumext_store_anskey_arg_tl`
`\l__enumext_store_anskey_env_tl`
`\l__enumext_store_anskey_opt_tl`
`\l__enumext_store_current_label_tl`
`\l__enumext_store_current_opt_arg_tl`
`\l__enumext_store_current_label_tmp_tl`

The variable `\l__enumext_store_active_bool` setting by save-ans key (§11.23.1) activates all the mechanism related to \anskey, anskeyenv, 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 (§11.26) and the variables `\l__enumext_store_anskey_env_tl`, `\l__enumext_store_anskey_opt_tl` save the ⟨*body*⟩ and ⟨*keys*⟩ of the environment anskeyenv (§11.27).

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* (§11.31.2) and \anspic* (§11.36.1) 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.

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

(End of definition for \l__enumext_store_active_bool and others.)

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

Internal variables used by the command \setenumext (§11.41).

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

(End of definition for \l__enumext_setkey_tmpa_tl and others.)

`\l__enumext_print_keyans_starred_tl`
`\l__enumext_mark_position_str`
`\g__enumext_item_symbol_tl`
`\l__enumext_print_keyans_X_tl`
`\l__enumext_store_save_key_X_tl`
`\l__enumext_store_save_key_X_bool`
`\l__enumext_store_upper_level_X_bool`

Internal variables used by command \printkeyans (§11.40), show-pos key (§11.25), item-sym* key (§11.29), save-key key (§11.25.2) and *"storage level system"*.

```
121 \tl_new:N  \l__enumext_print_keyans_starred_tl
122 \str_new:N \l__enumext_mark_position_str
123 \tl_new:N  \g__enumext_item_symbol_tl
124 \cs_set_protected:Npn \__enumext_tmp:n #1
125   {
126     \tl_new:c   { l__enumext_print_keyans_#1_tl         }
127     \tl_new:c   { l__enumext_store_save_key_#1_tl       }
128     \bool_new:c { l__enumext_store_save_key_#1_bool     }
129     \bool_new:c { l__enumext_store_upper_level_#1_bool }
130   }
131 \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.)

`\l__enumext_keyans_pic_body_seq`
`\l__enumext_keyans_pic_width_dim`
`\l__enumext_keyans_pic_above_int`
`\l__enumext_keyans_pic_below_int`
`\l__enumext_keyans_pic_above_skip`

Internal variables used by keyanspic environment (§11.36.2).

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

(End of definition for \l__enumext_keyans_pic_body_seq and others.)

`\l__enumext_check_answers_bool`
`\g__enumext_check_ans_key_bool`
`\l__enumext_check_start_line_env_tl`
`\g__enumext_check_starred_cmd_int`
`\g__enumext_item_anskey_int`
`\g__enumext_item_number_int`

Internal variables used by *"check answer"* mechanism (§11.23.3) 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.

```
137 \bool_new:N \l__enumext_check_answers_bool
138 \bool_new:N \g__enumext_check_ans_key_bool
139 \tl_new:N   \l__enumext_check_start_line_env_tl
140 \int_new:N  \g__enumext_check_starred_cmd_int
141 \int_new:N  \g__enumext_item_anskey_int
142 \int_new:N  \g__enumext_item_number_int
143 \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.

```
144 \bool_new:N \l__enumext_hyperref_bool
145 \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 (§11.25). 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.

```
146 \tl_new:N \l__enumext_newlabel_arg_one_tl
147 \tl_new:N \l__enumext_newlabel_arg_two_tl
148 \tl_new:N \l__enumext_write_aux_file_tl
149 \cs_set_protected:Npn \__enumext_tmp:n #1
150    {
151       \tl_new:c { l__enumext_label_copy_#1_tl }
152    }
153 \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` (§11.30).

```
154 \int_new:N \g__enumext_footnote_int
155 \seq_new:N \g__enumext_footnote_arg_seq
156 \seq_new:N \g__enumext_footnote_int_seq
```

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

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

Internal variables used by enumext* and keyans* environments.

```
157 \cs_set_protected:Npn \__enumext_tmp:n #1
158    {
159       \bool_new:c { l__enumext_item_starred_#1_bool   }
160       \int_new:c  { l__enumext_item_column_pos_#1_int }
161       \int_new:c  { g__enumext_item_count_all_#1_int  }
162       \int_new:c  { l__enumext_joined_item_#1_int     }
163       \int_new:c  { l__enumext_joined_item_aux_#1_int }
164       \int_new:c  { l__enumext_tmpa_#1_int            }
165       \box_new:c  { l__enumext_item_text_#1_box       }
166       \dim_new:c  { l__enumext_joined_width_#1_dim    }
167       \dim_new:c  { l__enumext_item_width_#1_dim      }
168       \tl_new:c   { g__enumext_item_symbol_aux_#1_tl  }
169       \str_new:c  { l__enumext_align_label_#1_str     }
170       \bool_new:c { g__enumext_minipage_active_#1_bool }
171       \tl_new:c   { g__enumext_miniright_code_#1_tl   }
172       \bool_new:c { g__enumext_minipage_center_#1_bool }
173       \dim_new:c  { g__enumext_minipage_right_#1_dim  }
174       \skip_new:c { g__enumext_minipage_right_#1_skip }
175    }
176 \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`.

```
177 \clist_const:Nn \c__enumext_all_envs_clist
178    {
179       {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv},
180       {keyans}{v}, {enumext*}{vii}, {keyans*}{viii}
181    }
```

(*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.

```
182  \cs_new_protected:Npn \__enumext_at_begin_document:n #1
183    {
184      \hook_gput_code:nnn {begindocument} {enumext} { #1 }
185    }
```

(*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-rigth` and `mini-rigth*` keys outside the `enumext*` and `keyans*` environments and print `check-ans` outside the `enumext` and `enumext*` environments.

```
186  \cs_new_protected:Npn \__enumext_after_env:nn #1 #2
187    {
188      \hook_gput_code:nnn {env/#1/after} {enumext} {#2}
189    }
190  \cs_new_protected:Npn \__enumext_before_env:nn #1 #2
191    {
192      \hook_gput_code:nnn {env/#1/before} {enumext} {#2}
193    }
```

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

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

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

(*End of definition for* __enumext_level:.)

__enumext_if_is_int:nT
__enumext_if_is_int:nF
__enumext_if_is_int:nTF A conditional function to know if the variable we are passing is an integer used by `start` and `widest` keys. This function is taken directly from the answer given by Henri Menke in How to test if an expl3 function argument is an integer expression?.

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

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

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

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

(*End of definition for* __enumext_regex_counter_style:.)

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

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

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

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.

```
218  \cs_new_protected:Nn \__enumext_is_not_nested:
219    {
220      \str_case:en { \@currenvir }
221        {
222          {enumext}
223            {
224              \bool_lazy_and:nnT
225                { \bool_not_p:n { \g__enumext_standar_bool } }
226                { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
227                {
228                  \bool_gset_true:N \g__enumext_standar_bool
229                }
230            }
231          {enumext*}
232            {
233              \bool_lazy_and:nnT
234                { \bool_not_p:n { \g__enumext_starred_bool } }
235                { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
236                {
237                  \bool_gset_true:N \g__enumext_starred_bool
238                }
239            }
240        }
241    }
```

The function `\__enumext_is_on_first_level:` will set the variables `\l__enumext_standar_first_bool` and `\l__enumext_starred_first_bool` 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.

```
242  \cs_new_protected:Nn \__enumext_is_on_first_level:
243    {
244      \bool_lazy_all:nT
245        {
246          { \bool_if_p:N \g__enumext_standar_bool }
247          { \int_compare_p:nNn { \l__enumext_level_int } = { 1 } }
248          { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
249        }
250        {
251          \bool_set_true:N \l__enumext_standar_first_bool
252          \tl_gset:Nn \g__enumext_envir_name_tl { enumext }
253          \tl_gset:Ne \g__enumext_start_line_tl
254            {
255              on ~ line ~ \exp_not:V \inputlineno
256            }
257        }
258      \bool_lazy_all:nT
259        {
260          { \bool_if_p:N \g__enumext_starred_bool }
261          { \int_compare_p:nNn { \l__enumext_level_h_int } = { 1 } }
262          { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
263        }
264        {
265          \bool_set_true:N \l__enumext_starred_first_bool
266          \tl_gset:Nn \g__enumext_envir_name_tl { enumext* }
267          \tl_gset:Ne \g__enumext_start_line_tl
268            {
269              on ~ line ~ \exp_not:V \inputlineno
270            }
271        }
272    }
```

(*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.

```
273  \cs_new_protected:Nn \__enumext_keyans_start_line:
274    {
275      \str_case:en { \@currenvir }
276        {
277          {keyans}
278            {
279              \tl_set:Ne \l__enumext_check_start_line_env_tl
280                {
281                  in ~ 'keyans' ~ start ~ on ~ line ~ \exp_not:V \inputlineno
282                }
283            }
284          {keyans*}
285            {
286              \tl_set:Ne \l__enumext_check_start_line_env_tl
287                {
288                  in ~ 'keyans*' ~ start ~ on ~ line ~ \exp_not:V \inputlineno
289                }
290            }
291          {keyanspic}
292            {
293              \tl_set:Ne \l__enumext_check_start_line_env_tl
294                {
295                  in ~ 'keyanspic' ~ start ~ on ~ line ~ \exp_not:V \inputlineno
296                }
297            }
298        }
299    }
```

(*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.

```
300  \cs_new_protected:Nn \__enumext_reset_global_vars:
301    {
302      \__enumext_reset_global_int:
303      \__enumext_reset_global_bool:
304      \__enumext_reset_global_tl:
305    }
306  \cs_new_protected:Nn \__enumext_reset_global_int:
307    {
308      \int_gzero:N \g__enumext_item_number_int
309      \int_gzero:N \g__enumext_item_anskey_int
310      \int_gzero:N \g__enumext_item_answer_diff_int
311    }
312  \cs_new_protected:Nn \__enumext_reset_global_bool:
313    {
314      \bool_gset_false:N \g__enumext_check_ans_key_bool
315      \bool_gset_false:N \g__enumext_standar_bool
316      \bool_gset_false:N \g__enumext_starred_bool
317    }
318  \cs_new_protected:Nn \__enumext_reset_global_tl:
319    {
320      \tl_gclear:N \g__enumext_store_name_tl
321      \tl_gclear:N \g__enumext_start_line_tl
322      \tl_gclear:N \g__enumext_envir_name_tl
323    }
```

(*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.

```
324  \cs_new_protected:Nn \__enumext_log_global_vars:
325    {
326      \msg_log:nneeee { enumext } { prop-seq-int-hook }
327        { \g__enumext_store_name_tl }
328        { \prop_count:c { g__enumext_ \g__enumext_store_name_tl _prop } }
329        { \seq_count:c { g__enumext_ \g__enumext_store_name_tl _seq } }
330        { \int_use:c { g__enumext_resume_ \g__enumext_store_name_tl _int } }
331    }
```

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.

```
332  \cs_new_protected:Nn \__enumext_log_answer_vars:
333    {
334      \msg_log:nneee { enumext } { item-answer-hook }
335        { \int_use:N \g__enumext_item_number_int }
336        { \int_use:N \g__enumext_item_anskey_int }
337        { \int_eval:n { \g__enumext_item_number_int - \g__enumext_item_anskey_int} }
338    }
```

(*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.

```
339  \__enumext_at_begin_document:n
340    {
341      \cs_new_eq:NN \__enumext_start_list:nn \list
342      \cs_new_eq:NN \__enumext_stop_list: \endlist
343      \cs_new_eq:NN \__enumext_item_std:w \item
344    }
```

(*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.

```
345  \__enumext_at_begin_document:n
346    {
347      \cs_new_eq:NN \__enumext_minipage:w \minipage
348      \cs_new_eq:NN \__enumext_endminipage: \endminipage
349    }
```

(*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 an internal `__enumext_mini_env*` environment (*custom version* of `minipage`) setting the `\if@minipage` switch to *"false"* to allow spaces at the *"above"* of the environment, plus we will add `\vspace{0pt}` to maintain alignment on *"top"*. This environment will be used internally by the `mini-env` key, it is not documented in the user interface and is for internal use only. This function is passed to the function `\__enumext_safe_exec:` in the `enumext` environment definition (§11.34) and `\__enumext_safe_exec_vii:` in the `enumext*` environment definition (§11.37)

```
350  \cs_new_protected:Nn \__enumext_internal_mini_page:
351    {
352      \int_compare:nNnT { \l__enumext_level_int } = { 0 }
353        {
354          \DeclareDocumentEnvironment{__enumext_mini_env*}{ m }
355            {
356              \__enumext_minipage:w [ t ] { ##1 }
357                \legacy_if_gset_false:n { @minipage }
358                \vspace { 0pt }
359            }
```

```
360        { \__enumext_endminipage: }
361      }
362    }
```

(*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.

```
363 \hook_gput_code:nnn { begindocument } { enumext } { \__enumext_after_hyperref: }
364 \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".

```
365 \cs_new_protected:Nn \__enumext_after_hyperref:
366   {
367     \IfPackageLoadedTF { hyperref }
368       {
369         \msg_info:nnn { enumext } { package-load } { hyperref }
370         \bool_set_true:N \l__enumext_hyperref_bool
371         \IfHyperBoolean{hyperfootnotes}
372           {
373             \typeout{hyperfootnotes=true}
374             \bool_set_true:N \l__enumext_footnotes_key_bool
375           }
376           { \typeout{hyperfootnotes=false} }
377       }
378       {  }
```

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.

```
379     \bool_if:NT \l__enumext_footnotes_key_bool
380       {
381         \IfPackageLoadedTF { footnotehyper }
382           {
383             \msg_info:nnn { enumext } { package-load } { footnotehyper }
384           }
385           {
386             \typeout{No ~ footnotehyper ~ load}
387             \typeout{Load ~ and  ~ use  ~ \string\makesavenoteenv{enumext*}}
388             \bool_set_false:N \l__enumext_footnotes_key_bool
389           }
390       }
```

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.

```
391     \bool_if:NTF \l__enumext_hyperref_bool
392       {
393         \cs_new_eq:NN \__enumext_hypertarget:nn \hypertarget
394         \cs_new_eq:NN \__enumext_phantomsection: \phantomsection
395       }
396       {
397         \cs_new_eq:NN \__enumext_hypertarget:nn \use_none:nn
398         \cs_new_eq:NN \__enumext_phantomsection: \prg_do_nothing:
399       }
400   }
```

(*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.

```
401  \cs_new_protected:Npn \__enumext_newlabel:nn #1 #2
402    {
403      \protected@write \@auxout { }
404        {
405          \token_to_str:N \newlabel {#1}
406            {
407              {#2}
408              \bool_if:NT \l__enumext_hyperref_bool
409                { { \thepage } {#2} {#1} }
410              { }
411            }
412        }
413      \__enumext_hypertarget:nn {#1} { }
414      \__enumext_phantomsection:
415    }
```

(*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.

```
416  \cs_new_protected:Npn \__enumext_define_counters:Nn #1 #2
417    {
418      \cs_if_exist:cTF { c@ #2 }
419        { \msg_fatal:nnn { enumext } { counters }{ #2 } }
420        {
421          \tl_set:Nn #1 { #2 }
422          \newcounter { #2 }
423        }
424    }
```

(*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.

```
425  \__enumext_define_counters:Nn \l__enumext_counter_i_tl    { enumXi     }
426  \__enumext_define_counters:Nn \l__enumext_counter_ii_tl   { enumXii    }
427  \__enumext_define_counters:Nn \l__enumext_counter_iii_tl  { enumXiii   }
428  \__enumext_define_counters:Nn \l__enumext_counter_iv_tl   { enumXiv    }
429  \__enumext_define_counters:Nn \l__enumext_counter_v_tl    { enumXv     }
430  \__enumext_define_counters:Nn \l__enumext_counter_vi_tl   { enumXvi    }
431  \__enumext_define_counters:Nn \l__enumext_counter_vii_tl  { enumXvii   }
432  \__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.

```
433  \cs_new_protected:Npn \__enumext_register_counter_style:Nn #1 #2
434    {
435      \tl_const:cn { c__enumext_widest_ \cs_to_str:N #1 _tl } {#2}
436      \tl_gput_right:Nn \g__enumext_counter_styles_tl {#1}
437    }
438  \__enumext_register_counter_style:Nn \arabic { 0 }
439  \__enumext_register_counter_style:Nn \Alph  { M }
440  \__enumext_register_counter_style:Nn \alph  { m }
441  \__enumext_register_counter_style:Nn \Roman { VIII }
442  \__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.

```
443 \cs_new_protected:Npn \__enumext_label_width_by_box:Nn #1 #2
444   {
445     \hbox_set:Nn \l__enumext_label_width_by_box {#2}
446     \dim_set:Nn #1 { \box_wd:N \l__enumext_label_width_by_box }
447   }
448 \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.

```
449 \cs_new_protected:Npn \__enumext_label_style:Nnn #1 #2 #3
450   {
451     \tl_clear_new:N #1
452     \tl_put_right:Ne #1 { \tl_trim_spaces:n {#3} }
453     \tl_gset_eq:NN \g__enumext_widest_label_tl #1
454     \tl_map_inline:Nn \g__enumext_counter_styles_tl
455       {
456         \tl_replace_all:Nne #1 { ##1* } { \exp_not:N ##1 {#2} }
457         \tl_greplace_all:Nne \g__enumext_widest_label_tl { ##1* }
458           { \tl_use:c { c__enumext_widest_ \cs_to_str:N ##1 _tl } }
459       }
460     \__enumext_label_width_by_box:Nn \l__enumext_current_widest_dim
461       { \tl_use:N \g__enumext_widest_label_tl }
462     \tl_set_eq:cN { the #2 } #1
463   }
464 \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.

```
465 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
466   {
467     \keys_define:nn { enumext / #1 }
468       {
469         font       .tl_set:c   = { l__enumext_label_font_style_#2_tl },
470         font       .value_required:n = true,
471         labelsep   .dim_set:c  = { l__enumext_labelsep_#2_dim },
472         labelsep   .initial:n  = {0.3333em},
473         labelsep   .value_required:n = true,
474         labelwidth .dim_set:c  = { l__enumext_labelwidth_#2_dim },
475         labelwidth .value_required:n = true,
476         wrap-label .cs_set_protected:cp = { __enumext_wrapper_label_#2:n } ##1,
477         wrap-label .initial:n  = {##1},
478         wrap-label .value_required:n = true,
479         wrap-label* .code:n = {
480                       \bool_set_true:c { l__enumext_wrap_label_opt_#2_bool }
481                       \keys_set:nn { enumext / #1 } { wrap-label = {##1} }
482                     },
483         wrap-label* .value_required:n = true,
484       }
485   }
486 \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.

```
487 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
488   {
489     \keys_define:nn { enumext / #1 }
490       {
491         align .choice:,
492         align / left   .code:n =
493                         {
494                           \tl_clear:c { l__enumext_label_fill_left_#2_tl  }
495                           \tl_set:cn  { l__enumext_label_fill_right_#2_tl } { \hfill }
496                         },
497         align / right  .code:n =
498                         {
499                           \tl_set:cn  { l__enumext_label_fill_left_#2_tl  } { \hfill }
500                           \tl_clear:c { l__enumext_label_fill_right_#2_tl }
501                         },
502         align / center .code:n =
503                         {
504                           \tl_set:cn { l__enumext_label_fill_left_#2_tl  } { \hfill }
505                           \tl_set:cn { l__enumext_label_fill_right_#2_tl } { \hfill }
506                         },
507         align .initial:n  = left,
508         align .value_required:n  = true,
509       }
510   }
511 \clist_map_inline:nn
512   {
513     {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {keyans}{v}
514   }
515   { \__enumext_tmp:nn #1 }

516 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
517   {
518     \keys_define:nn { enumext / #1 }
519       {
520         align .choice:,
521         align / left   .code:n = \str_set:cn { l__enumext_align_label_#2_str } { l },
522         align / right  .code:n = \str_set:cn { l__enumext_align_label_#2_str } { r },
523         align / center .code:n = \str_set:cn { l__enumext_align_label_#2_str } { c },
524         align .initial:n = left,
525         align .value_required:n = true,
526       }
527   }
528 \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.

```
529 \cs_set_protected:Npn \__enumext_tmp:nnn #1 #2 #3
530   {
531     \keys_define:nn { enumext / #1 }
532       {
533         label .code:n     = {
534                             \__enumext_label_style:cvn { l__enumext_label_#2_tl }
535                               { l__enumext_counter_#2_tl } { ##1}
536                             \dim_set_eq:cN  { l__enumext_labelwidth_#2_dim }
537                               \l__enumext_current_widest_dim
538                           },
539         label .initial:n = #3,
540         label .value_required:n = true,
541         ref   .code:n     = \__enumext_standar_ref:n {##1},
542         ref   .value_required:n = true,
543       }
```

```
544    }
545  \__enumext_tmp:nnn { level-1 } {   i } { \arabic*.}
546  \__enumext_tmp:nnn { level-2 } {  ii } { (\alph*) }
547  \__enumext_tmp:nnn { level-3 } { iii } { \roman*. }
548  \__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.

```
549  \cs_new_protected:Npn \__enumext_standar_ref:n #1
550    {
551      \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
552      \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
553        {
554          \msg_error:nnn { enumext } { key-ref-empty } { enumext }
555        }
556        {
557          \tl_set_eq:Nc
558            \l__enumext_ref_the_count_tl { l__enumext_counter_ \__enumext_level: _tl }
559          \__enumext_regex_counter_style:
560          \tl_set_eq:Nc
561            \l__enumext_ref_the_count_tl { l__enumext_the_counter_ \__enumext_level: _tl }
562          \tl_put_right:ce { l__enumext_renew_the_count_ \__enumext_level: _tl }
563            {
564              \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl }
565                { \exp_not:V \l__enumext_ref_key_arg_tl }
566            }
567        }
568    }
```

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

```
569  \cs_new_protected:Nn \__enumext_standar_ref:
570    {
571      \tl_if_empty:cF { l__enumext_renew_the_count_ \__enumext_level: _tl }
572        {
573          \tl_use:c { l__enumext_renew_the_count_ \__enumext_level: _tl }
574        }
575    }
```

(*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`
`\l__enumext_label_vii_tl`
`\l__enumext_label_viii_tl`

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

```
576  \cs_set_protected:Npn \__enumext_tmp:nnn #1 #2 #3
577    {
578      \keys_define:nn { enumext / #1 }
579        {
580          label .code:n     = {
581                               \__enumext_label_style:cvn { l__enumext_label_#2_tl }
582                                 { l__enumext_counter_#2_tl } {##1}
583                               \dim_set_eq:cN  { l__enumext_labelwidth_#2_dim }
584                                 \l__enumext_current_widest_dim
585                             },
586          label .initial:n = #3,
587          label .value_required:n = true,
588          ref    .code:n     = \__enumext_starred_ref:n {##1},
589          ref    .value_required:n = true,
590        }
591    }
592  \__enumext_tmp:nnn { enumext* } {  vii } { \arabic*.}
593  \__enumext_tmp:nnn { keyans*  } { viii } { \Alph*) }
```

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

__enumext_starred_ref:n
__enumext_starred_ref:

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

```
594 \cs_new_protected:Npn \__enumext_starred_ref:n #1
595   {
596     \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
597     \int_compare:nNnT { \l__enumext_level_h_int } = { 1 }
598       {
599         \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
600           {
601             \msg_error:nnn { enumext } { key-ref-empty } { enumext* }
602           }
603           {
604             \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_counter_vii_tl
605             \__enumext_regex_counter_style:
606             \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_the_counter_vii_tl
607             \tl_put_right:Ne \l__enumext_renew_the_count_vii_tl
608               {
609                 \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl }
610                   { \exp_not:V \l__enumext_ref_key_arg_tl }
611               }
612           }
613       }
614     \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
615       {
616         \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
617           {
618             \msg_error:nnn { enumext } { key-ref-empty } { keyans* }
619           }
620           {
621             \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_counter_viii_tl
622             \__enumext_regex_counter_style:
623             \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_the_counter_viii_tl
624             \tl_put_right:Ne \l__enumext_renew_the_count_viii_tl
625               {
626                 \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl }
627                   { \exp_not:V \l__enumext_ref_key_arg_tl }
628               }
629           }
630       }
631   }
```

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.

```
632 \cs_new_protected:Nn \__enumext_starred_ref:
633   {
634     \int_compare:nNnT { \l__enumext_level_h_int } = { 1 }
635       {
636         \tl_if_empty:NF \l__enumext_renew_the_count_vii_tl
637           {
638             \tl_use:N \l__enumext_renew_the_count_vii_tl
639           }
640       }
641     \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
642       {
643         \tl_if_empty:NF \l__enumext_renew_the_count_viii_tl
644           {
645             \tl_use:N \l__enumext_renew_the_count_viii_tl
646           }
647       }
648   }
```

(*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.

```
649 \keys_define:nn { enumext / keyans }
650   {
651     label .code:n    = {
652                         \__enumext_label_style:cvn { l__enumext_label_v_tl }
653                           { l__enumext_counter_v_tl } {#1}
```

```
654                            \dim_set_eq:cN  { l__enumext_labelwidth_v_dim }
655                              \l__enumext_current_widest_dim
656                            \__enumext_label_style:cvn { l__enumext_label_vi_tl }
657                              { l__enumext_counter_vi_tl } {#1}
658                            \dim_set_eq:cN  { l__enumext_labelwidth_v_dim }
659                              \l__enumext_current_widest_dim
660                          },
661        label .initial:n = \Alph*),
662        label .value_required:n = true,
663        ref    .code:n    = \__enumext_keyans_ref:n {#1},
664        ref    .value_required:n = true,
665      }
```

(*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.

```
666 \cs_new_protected:Npn \__enumext_keyans_ref:n #1
667    {
668      \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
669      \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
670        {
671          \msg_error:nnn { enumext } { key-ref-empty } { keyans }
672        }
673        {
674          \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_counter_v_tl
675          \__enumext_regex_counter_style:
676          \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_the_counter_v_tl
677          \tl_put_right:Ne \l__enumext_renew_the_count_v_tl
678            {
679              \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl }
680                { \exp_not:V \l__enumext_ref_key_arg_tl }
681            }
682        }
683    }
```

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

```
684 \cs_new_protected:Nn \__enumext_keyans_ref:
685    {
686      \tl_if_empty:NF \l__enumext_renew_the_count_v_tl
687        {
688          \tl_use:N \l__enumext_renew_the_count_v_tl
689        }
690    }
```

(*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.

```
691 \cs_new_protected:Npn \__enumext_start_from:NNn #1 #2 #3
692    {
693      \__enumext_if_is_int:nTF { #3 }
694        {
695          \int_set:Nn #2 {#3}
696        }
697        {
698          \regex_match:nVT { \c{Alph} | \c{alph} } {#1}
699            { \int_set:Nn #2 { \int_from_alph:n {#3} } }
700          \regex_match:nVT { \c{Roman} | \c{roman} } {#1}
701            { \int_set:Nn #2  { \int_from_roman:n {#3} } }
702        }
703    }
704 \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.

```
705 \cs_new_protected:Npn \__enumext_widest_from:nNNn #1 #2 #3 #4
706   {
707     \__enumext_if_is_int:nTF {#4}
708       {
709         \setcounter{enumX#1} { #4 }
710       }
711       {
712         \regex_match:nVT { \c{Alph} | \c{alph} } {#2}
713           { \setcounter{enumX#1} { \int_from_alph:n {#4} } }
714         \regex_match:nVT { \c{Roman} | \c{roman} } {#2}
715           { \setcounter{enumX#1} { \int_from_roman:n {#4} } }
716       }
717     \__enumext_label_width_by_box:cv
718       { l__enumext_labelwidth_#1_dim } { l__enumext_label_#1_tl }
719   }
720 \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.

```
721 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
722   {
723     \keys_define:nn { enumext / #1 }
724       {
725         start  .code:n    = {
726                             \__enumext_start_from:ccn
727                               { l__enumext_label_#2_tl }
728                               { l__enumext_start_#2_int } {##1}
729                             },
730         start  .initial:n = 1,
731         widest .code:n    = {
732                             \__enumext_widest_from:nccn {#2}
733                               { l__enumext_label_#2_tl }
734                               { l__enumext_labelwidth_#2_dim } {##1}
735                             },
736         widest .value_required:n = true,
737         start  .value_required:n = true,
738       }
739   }
740 \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.

```
741 \cs_set_protected:Npn \__enumext_tmp:nnnnnn #1 #2 #3 #4 #5 #6
742   {
743     \keys_define:nn { enumext / #1 }
744       {
745         topsep    .skip_set:c = { l__enumext_topsep_#2_skip },
746         topsep    .initial:n  = {#3},
747         topsep    .value_required:n = true,
748         partopsep .skip_set:c = { l__enumext_partopsep_#2_skip },
749         partopsep .initial:n  = {#4},
750         partopsep .value_required:n = true,
751         parsep    .skip_set:c = { l__enumext_parsep_#2_skip },
752         parsep    .initial:n  = {#5},
```

```
753        parsep    .value_required:n = true,
754        itemsep   .skip_set:c = { l__enumext_itemsep_#2_skip },
755        itemsep   .initial:n  = {#6},
756        itemsep   .value_required:n = true,
757        noitemsep .meta:n      = { itemsep = 0pt, parsep = 0pt },
758        noitemsep .value_forbidden:n = true,
759        nosep     .meta:n      = {
760                                   itemsep = 0pt, parsep= 0pt,
761                                   topsep = 0pt, partopsep = 0pt,
762                                 },
763        nosep     .value_forbidden:n = true,
764      }
765    }
```

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

```
766 \__enumext_tmp:nnnnnn { level-1 } { i } { 8.0pt plus 2.0pt minus 4.0pt }
767   { 2.0pt plus 1.0pt minus 1.0pt } { 4.0pt plus 2.0pt minus 1.0pt }
768   { 4.0pt plus 2.0pt minus 1.0pt }
769 \__enumext_tmp:nnnnnn { level-2 } { ii } { 4.0pt plus 2.0pt minus 1.0pt }
770   { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
771   { 2.0pt plus 1.0pt minus 1.0pt }
772 \__enumext_tmp:nnnnnn { level-3 } { iii } { 2.0pt plus 1.0pt minus 1.0pt }
773   { 1.0pt minus 1.0pt }{ 0pt }{ 2.0pt plus 1.0pt minus 1.0pt }
774 \__enumext_tmp:nnnnnn { level-4 } { iv } { 2.0pt plus 1.0pt minus 1.0pt }
775   { 1.0pt minus 1.0pt }{ 0pt }{ 2.0pt plus 1.0pt minus 1.0pt }
776 \__enumext_tmp:nnnnnn { keyans  } { v }{ 4.0pt plus 2.0pt minus 1.0pt }
777   { 2.0pt plus 1.0pt minus 1.0pt }{ 2.0pt plus 1.0pt minus 1.0pt }
778   { 2.0pt plus 1.0pt minus 1.0pt }
779 \__enumext_tmp:nnnnnn { enumext* } { vii } { 8.0pt plus 2.0pt minus 4.0pt }
780   { 2.0pt plus 1.0pt minus 1.0pt } { 4.0pt plus 2.0pt minus 1.0pt }
781   { 4.0pt plus 2.0pt minus 1.0pt }
782 \__enumext_tmp:nnnnnn { keyans* } { viii } { 4.0pt plus 2.0pt minus 1.0pt }
783   { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
784   { 2.0pt plus 1.0pt minus 1.0pt }
```

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

11.15   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.

```
785 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
786   {
787     \keys_define:nn { enumext / #1 }
788       {
789         itemindent    .dim_set:c = { l__enumext_fake_item_indent_#2_dim },
790         itemindent    .value_required:n = true,
791         rightmargin   .dim_set:c = { l__enumext_rightmargin_#2_dim },
792         rightmargin   .value_required:n = true,
793         listparindent .dim_set:c = { l__enumext_listparindent_#2_dim },
794         listparindent .value_required:n = true,
795         list-offset   .dim_set:c = { l__enumext_listoffset_#2_dim },
796         list-offset   .value_required:n = true,
797         list-indent   .code:n    =
798                         \bool_set_true:c { l__enumext_leftmargin_tmp_#2_bool }
799                         \dim_set:cn { l__enumext_leftmargin_tmp_#2_dim } {##1},
800         list-indent   .value_required:n = true,
801       }
802   }
803 \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.

```
804 \cs_set_protected:Npn \__enumext_tmp:n #1
805   {
806     \keys_define:nn { enumext / #1 } { list-indent .initial:n  = 0pt, }
807   }
808 \clist_map_inline:nn { enumext*, keyans* } { \__enumext_tmp:n {#1} }
```

11.15.1 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.

```
809  \cs_set_protected:Nn \__enumext_fake_item:
810    {
811      \dim_compare:nNnT
812        { \dim_use:c { l__enumext_fake_item_indent_ \__enumext_level: _dim } }
813        >
814        { \c_zero_dim }
815        {
816          \tl_set:ce { l__enumext_fake_item_indent_ \__enumext_level: _tl }
817            {
818              \exp_not:N \mode_leave_vertical:
819              \exp_not:n { \skip_horizontal:n }
820                { \dim_use:c { l__enumext_fake_item_indent_ \__enumext_level: _dim } }
821              \ignorespaces
822            }
823        }
824    }
825  \cs_set_protected:Nn \__enumext_keyans_fake_item:
826    {
827      \dim_compare:nNnT
828        { \l__enumext_fake_item_indent_v_dim } > { \c_zero_dim }
829        {
830          \tl_set:Ne \l__enumext_fake_item_indent_v_tl
831            {
832              \exp_not:N \mode_leave_vertical:
833              \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_v_dim
834            }
835        }
836    }
837  \cs_set_protected:Nn \__enumext_fake_item_vii:
838    {
839      \dim_compare:nNnT
840        { \l__enumext_fake_item_indent_vii_dim } > { \c_zero_dim }
841        {
842          \tl_set:Ne \l__enumext_fake_item_indent_vii_tl
843            {
844              \exp_not:N \mode_leave_vertical:
845              \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_vii_dim
846            }
847        }
848    }
849  \cs_set_protected:Nn \__enumext_fake_item_viii:
850    {
851      \dim_compare:nNnT
852        { \l__enumext_fake_item_indent_viii_dim } > { \c_zero_dim }
853        {
854          \tl_set:Ne \l__enumext_fake_item_indent_viii_tl
855            {
856              \exp_not:N \mode_leave_vertical:
857              \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_viii_dim
858            }
859        }
860    }
```

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

11.16 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.

```
861  \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
862    {
863      \keys_define:nn { enumext / #1 }
864        {
865          show-length .bool_set:c = { l__enumext_show_length_#2_bool },
```

```
866        show-length .initial:n  = false,
867      }
868    }
869  \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }
```

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

11.17 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.

```
870  \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
871    {
872      \keys_define:nn { enumext / #1 }
873        {
874          before  .tl_set:c   = { l__enumext_before_no_starred_key_#2_tl },
875          before  .value_required:n = true,
876          before* .tl_set:c   = { l__enumext_before_starred_key_#2_tl },
877          before* .value_required:n = true,
878          after   .tl_set:c   = { l__enumext_after_stop_list_#2_tl },
879          after   .value_required:n = true,
880          first   .tl_set:c   = { l__enumext_after_list_args_#2_tl },
881          first   .value_required:n = true,
882        }
883    }
884  \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }
```

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

11.17.1 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.

```
885  \cs_new_protected:Nn \__enumext_before_args_exec:
886    {
887      \tl_use:c { l__enumext_before_starred_key_ \__enumext_level: _tl }
888    }
```

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*⟩.

```
889  \cs_new_protected:Nn \__enumext_before_keys_exec:
890    {
891      \tl_use:c { l__enumext_before_no_starred_key_ \__enumext_level: _tl }
892    }
```

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

```
893  \cs_new_protected:Nn \__enumext_after_stop_list:
894    {
895      \tl_use:c { l__enumext_after_stop_list_ \__enumext_level: _tl }
896    }
```

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.

```
897  \cs_new_protected:Nn \__enumext_after_args_exec:
898    {
899      \tl_use:c { l__enumext_after_list_args_ \__enumext_level: _tl }
900    }
```

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

11.17.2 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.

```
901  \cs_new_protected:Nn \__enumext_before_args_exec_v:
902    {
903      \tl_use:N \l__enumext_before_starred_key_v_tl
904    }
```

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*⟩.

```
905 \cs_new_protected:Nn \__enumext_before_keys_exec_v:
906    {
907       \tl_use:N \l__enumext_before_no_starred_key_v_tl
908    }
```

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

```
909 \cs_new_protected:Nn \__enumext_after_stop_list_v:
910    {
911       \tl_use:N \l__enumext_after_stop_list_v_tl
912    }
```

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.

```
913 \cs_new_protected:Nn \__enumext_after_args_exec_v:
914    {
915       \tl_use:N \l__enumext_after_list_args_v_tl
916    }
```

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

11.17.3 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.

```
917 \cs_new_protected:Nn \__enumext_before_args_exec_vii:
918    {
919       \tl_use:N \l__enumext_before_starred_key_vii_tl
920    }
921 \cs_new_protected:Nn \__enumext_before_args_exec_viii:
922    {
923       \tl_use:N \l__enumext_before_starred_key_viii_tl
924    }
```

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*⟩.

```
925 \cs_new_protected:Nn \__enumext_before_keys_exec_vii:
926    {
927       \tl_use:N \l__enumext_before_no_starred_key_vii_tl
928    }
929 \cs_new_protected:Nn \__enumext_before_keys_exec_viii:
930    {
931       \tl_use:N \l__enumext_before_no_starred_key_viii_tl
932    }
```

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

```
933 \cs_new_protected:Nn \__enumext_after_stop_list_vii:
934    {
935       \tl_use:N \l__enumext_after_stop_list_vii_tl
936    }
937 \cs_new_protected:Nn \__enumext_after_stop_list_viii:
938    {
939       \tl_use:N \l__enumext_after_stop_list_viii_tl
940    }
```

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.

```
941 \cs_new_protected:Nn \__enumext_after_args_exec_vii:
942    {
943       \tl_use:N \l__enumext_after_list_args_vii_tl
944    }
945 \cs_new_protected:Nn \__enumext_after_args_exec_viii:
946    {
947       \tl_use:N \l__enumext_after_list_args_viii_tl
948    }
```

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

11.18 Setting keys for `multicols` and `minipage`

<div style="float:left">mini-env
mini-sep
columns-sep
columns</div>

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.

```
949  \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
950    {
951      \keys_define:nn { enumext / #1 }
952        {
953          mini-env    .dim_set:c  = { l__enumext_minipage_right_#2_dim },
954          mini-env    .value_required:n = true,
955          mini-sep    .dim_set:c  = { l__enumext_minipage_hsep_#2_dim },
956          mini-sep    .initial:n  = 0.3333em,
957          mini-sep    .value_required:n = true,
958          columns-sep .dim_set:c  = { l__enumext_columns_sep_#2_dim },
959          columns-sep .value_required:n = true,
960          columns     .int_set:c  = { l__enumext_columns_#2_int },
961          columns     .initial:n  = 1,
962          columns     .value_required:n = true,
963        }
964    }
965  \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.

```
966  \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
967    {
968      \keys_define:nn { enumext / #1 }
969        {
970          mini-right  .tl_gset:c = { g__enumext_miniright_code_#2_tl },
971          mini-right  .value_required:n = true,
972          mini-right* .code:n    = {
973                                    \bool_gset_true:c { g__enumext_minipage_center_#2_bool }
974                                    \keys_set:nn { enumext / #1 } { miniright = {##1} }
975                                  },
976          mini-right* .value_required:n = true,
977        }
978    }
979  \clist_map_inline:nn { {enumext*}{vii}, {keyans*}{viii} } { \__enumext_tmp:nn #1 }
```

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

11.19 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.

11.19.1 Adjustment of vertical spaces for `multicols` in `enumext`

<div style="float:left">__enumext_multi_set_vskip:</div>

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*.

```
980  \cs_new_protected:Nn \__enumext_multi_set_vskip:
981    {
982      \skip_set:cn { l__enumext_multicols_above_ \__enumext_level: _skip }
983        {
984          \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
985        }
986      \skip_set:cn { l__enumext_multicols_below_ \__enumext_level: _skip }
987        {
988          \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
989        }
990      \__enumext_add_pre_parsep:
991    }
```

(*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*.

```
992   \cs_new_protected:Nn \__enumext_add_pre_parsep:
993     {
994       \int_case:nn { \l__enumext_level_int }
995         {
996           { 2 }{
997               \skip_if_eq:nnF { \l__enumext_parsep_i_skip } { \c_zero_skip }
998                 {
999                   \skip_add:Nn \l__enumext_multicols_above_ii_skip { \l__enumext_parsep_i_skip }
1000                 }
1001             }
1002           { 3 }{
1003               \skip_if_eq:nnF { \l__enumext_parsep_ii_skip } { \c_zero_skip }
1004                 {
1005                   \skip_add:Nn \l__enumext_multicols_above_iii_skip { \l__enumext_parsep_ii_skip }
1006                 }
1007             }
1008           { 4 }{
1009               \skip_if_eq:nnF { \l__enumext_parsep_iii_skip } { \c_zero_skip }
1010                 {
1011                   \skip_add:Nn \l__enumext_multicols_above_iv_skip { \l__enumext_parsep_iii_skip }
1012                 }
1013             }
1014         }
1015     }
```

(*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*⟩.

```
1016  \cs_new_protected:Nn \__enumext_multi_addvspace:
1017    {
1018      \__enumext_multi_set_vskip:
1019      \mode_if_vertical:T
1020        {
1021          \skip_add:cn { l__enumext_multicols_above_ \__enumext_level: _skip }
1022            {
1023              \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
1024            }
1025          \skip_add:cn { l__enumext_multicols_below_ \__enumext_level: _skip }
1026            {
1027              \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
1028            }
1029        }
1030      \par\nopagebreak
1031      \addvspace{ \skip_use:c { l__enumext_multicols_above_ \__enumext_level: _skip } }
1032    }
```

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

11.19.2 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`.

```
1033 \cs_new_protected:Nn \__enumext_keyans_multi_set_vskip:
1034   {
1035     \skip_set:Nn \l__enumext_multicols_above_v_skip
1036       {
1037         \l__enumext_topsep_v_skip
1038       }
1039     \skip_set:Nn \l__enumext_multicols_below_v_skip
1040       {
1041         \l__enumext_topsep_v_skip
1042       }
1043   }
1044 \cs_new_protected:Nn \__enumext_keyans_multi_addvspace:
1045   {
1046     \__enumext_keyans_multi_set_vskip:
1047     \mode_if_vertical:T
1048       {
1049         \skip_add:Nn \l__enumext_multicols_above_v_skip
1050           {
1051             \skip_use:N \l__enumext_partopsep_v_skip
1052           }
1053         \skip_add:Nn \l__enumext_multicols_below_v_skip
1054           {
1055             \skip_use:N \l__enumext_partopsep_v_skip
1056           }
1057       }
1058     \par\nopagebreak
1059     \addvspace{ \l__enumext_multicols_above_v_skip }
1060   }
```

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

11.20 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.
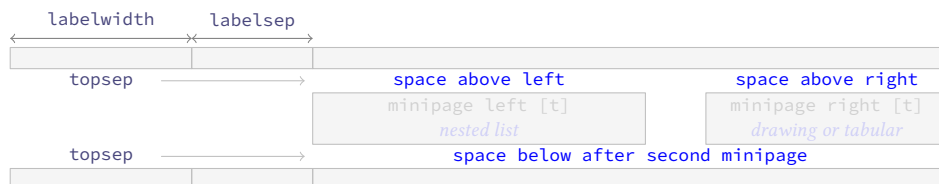
Figure 8: Representation of the `minipage` spacing adjustment for a nested level.

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`[13] package, again my attempts to find the correct values using `\showoutput` and `\showboxdepth` absolutely failed.

11.20.1 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.

```
1061  \cs_new_protected:Nn \__enumext_mini_set_vskip:
1062    {
1063      \int_compare:nNnTF
1064        { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } > { 1 }
1065        {
```

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.

```
1066          \skip_if_eq:nnTF
1067            { \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip } } { \c_zero_skip }
1068            {
1069              \skip_set:Nn \l__enumext_minipage_left_skip
1070                {
1071                  -0.150\box_dp:N \strutbox
1072                }
1073              \skip_set:Nn \l__enumext_minipage_right_skip
1074                {
1075                  0.695\box_dp:N \strutbox
1076                }
1077              \skip_set:Nn \l__enumext_minipage_after_skip
1078                {
1079                  \box_dp:N \strutbox
1080                }
1081              \__enumext_zero_parsep:
1082            }
1083            {
1084              \skip_set:Nn \l__enumext_minipage_left_skip
1085                {
1086                  \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
1087                }
1088              \skip_set:Nn \l__enumext_minipage_right_skip
1089                {
1090                  0.695\box_dp:N \strutbox
1091                }
1092              \skip_set:Nn \l__enumext_minipage_after_skip
1093                {
1094                  1.85\box_dp:N \strutbox
1095                  + \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
1096                }
1097            }
1098        }
1099        {
```

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.

```
1100          \skip_if_eq:nnTF
1101            { \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip } } { \c_zero_skip }
1102            {
1103              \skip_set:Nn \l__enumext_minipage_left_skip
1104                {
1105                  0.5\box_dp:N \strutbox
1106                  - \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
1107                }
1108              \skip_set:Nn \l__enumext_minipage_right_skip
1109                {
1110                  \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
1111                }
1112              \skip_set:Nn \l__enumext_minipage_after_skip
1113                {
1114                  1.6\box_dp:N \strutbox
1115                }
1116            }
1117            {
1118              \skip_set:Nn \l__enumext_minipage_left_skip
1119                {
1120                  0.5875\box_dp:N \strutbox
1121                  - \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
```

```
1122              }
1123          \skip_set:Nn \l__enumext_minipage_right_skip
1124            {
1125              + \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
1126              + \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
1127            }
1128          \skip_set:Nn \l__enumext_minipage_after_skip
1129            {
1130              0.325\box_dp:N \strutbox
1131              + \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
1132            }
1133          }
1134        }
1135    }
```

(*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.

```
1136  \cs_new_protected:Nn \__enumext_zero_parsep:
1137    {
1138      \int_case:nn { \l__enumext_level_int }
1139        {
1140          { 2 }{
1141              \skip_if_eq:nnF { \l__enumext_parsep_i_skip } { \c_zero_skip }
1142                {
1143                  \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
1144                }
1145            }
1146          { 3 }{
1147              \skip_if_eq:nnF { \l__enumext_parsep_ii_skip } { \c_zero_skip }
1148                {
1149                  \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
1150                }
1151            }
1152          { 4 }{
1153              \skip_if_eq:nnF { \l__enumext_parsep_iii_skip } { \c_zero_skip }
1154                {
1155                  \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
1156                }
1157            }
1158        }
1159    }
```

(*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*.

```
1160  \cs_new_protected:Nn \__enumext_mini_addvspace:
1161    {
1162      \__enumext_mini_set_vskip:
1163      \mode_if_vertical:T
1164        {
1165          \skip_add:Nn \l__enumext_minipage_left_skip
1166            {
1167              \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
1168            }
1169          \skip_add:Nn \l__enumext_minipage_after_skip
1170            {
1171              \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
1172            }
1173        }
1174      \par\nopagebreak
1175      \addvspace { \l__enumext_minipage_left_skip }
1176    }
```

(*End of definition for* __enumext_mini_addvspace:.)

11.20.2 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.

```
1177 \cs_new_protected:Nn \__enumext_keyans_mini_set_vskip:
1178   {
1179     \skip_zero_new:N \l__enumext_minipage_after_skip
1180     \skip_zero_new:N \l__enumext_minipage_left_skip
1181     \skip_zero_new:N \l__enumext_minipage_right_skip
1182     \int_compare:nNnTF { \l__enumext_columns_v_int } > { 1 }
1183       {
1184         \skip_if_eq:nnTF { \l__enumext_topsep_v_skip } { \c_zero_skip }
1185           {
1186             \skip_set:Nn \l__enumext_minipage_left_skip { -0.25\box_dp:N \strutbox }
1187             \skip_set:Nn \l__enumext_minipage_right_skip { 0.705\box_dp:N \strutbox }
1188             \skip_set:Nn \l__enumext_minipage_after_skip { \box_dp:N \strutbox }
1189             \skip_if_eq:nnF { \l__enumext_parsep_i_skip } { \c_zero_skip }
1190               {
1191                 \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
1192               }
1193           }
1194           {
1195             \skip_set:Nn \l__enumext_minipage_left_skip
1196               {
1197                 \skip_use:N \l__enumext_topsep_v_skip
1198               }
1199             \skip_set:Nn \l__enumext_minipage_right_skip
1200               {
1201                 0.705\box_dp:N \strutbox
1202               }
1203             \skip_set:Nn \l__enumext_minipage_after_skip
1204               {
1205                 1.85\box_dp:N \strutbox + \l__enumext_topsep_v_skip
1206               }
1207           }
1208       }
1209       {
1210         \skip_if_eq:nnTF { \l__enumext_topsep_v_skip } { \c_zero_skip }
1211           {
1212             \skip_set:Nn \l__enumext_minipage_left_skip
1213               {
1214                 0.5\box_dp:N \strutbox
1215                 + \l__enumext_partopsep_v_skip
1216               }
1217             \skip_set:Nn \l__enumext_minipage_right_skip
1218               {
1219                 \l__enumext_partopsep_v_skip
1220               }
1221             \skip_set:Nn \l__enumext_minipage_after_skip { 1.6\box_dp:N \strutbox }
1222           }
1223           {
1224             \skip_set:Nn \l__enumext_minipage_left_skip
1225               {
1226                 0.5875\box_dp:N \strutbox - \l__enumext_partopsep_v_skip
1227               }
1228             \skip_set:Nn \l__enumext_minipage_right_skip
1229               {
1230                 \l__enumext_topsep_v_skip + \l__enumext_partopsep_v_skip
1231               }
1232             \skip_set:Nn \l__enumext_minipage_after_skip
1233               {
1234                 0.325\box_dp:N \strutbox + \l__enumext_topsep_v_skip
1235               }
1236           }
1237       }
1238   }
```

(*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.

```
1239 \cs_new_protected:Nn \__enumext_keyans_mini_addvspace:
1240   {
1241     \__enumext_keyans_mini_set_vskip:
1242     \mode_if_vertical:T
1243       {
1244         \skip_add:Nn \l__enumext_minipage_left_skip
1245           {
1246             \l__enumext_partopsep_v_skip
1247           }
1248         \skip_add:Nn \l__enumext_minipage_after_skip
1249           {
1250             \l__enumext_partopsep_v_skip
1251           }
1252       }
1253     \par\nopagebreak
1254     \addvspace { \l__enumext_minipage_left_skip }
1255   }
```

(*End of definition for* __enumext_keyans_mini_addvspace:.)

11.20.3 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*.

```
1256 \cs_new_protected:Nn \__enumext_mini_set_vskip_vii:
1257   {
1258     \skip_zero_new:N \l__enumext_minipage_left_skip
1259     \skip_gzero_new:N \g__enumext_minipage_right_skip
1260     \skip_gzero_new:N \g__enumext_minipage_after_skip
1261     \skip_if_eq:nnTF { \l__enumext_topsep_vii_skip } { \c_zero_skip }
1262       {
1263         \skip_set:Nn \l__enumext_minipage_left_skip { 0.5\box_dp:N \strutbox }
1264         \skip_gset:Nn \g__enumext_minipage_right_skip { 0.325\box_dp:N \strutbox }
1265       }
1266       {
1267         \skip_set:Nn \l__enumext_minipage_left_skip { 0.5875\box_dp:N \strutbox }
1268         \skip_gset:Nn \g__enumext_minipage_right_skip
1269           {
1270             \l__enumext_topsep_vii_skip
1271           }
1272         \skip_gset:Nn \g__enumext_minipage_after_skip
1273           {
1274             0.325\box_dp:N \strutbox + \l__enumext_topsep_vii_skip
1275           }
1276       }
1277   }
1278 \cs_new_protected:Nn \__enumext_mini_set_vskip_viii:
1279   {
1280     \skip_zero_new:N \l__enumext_minipage_after_skip
1281     \skip_zero_new:N \l__enumext_minipage_left_skip
1282     \skip_zero_new:N \l__enumext_minipage_right_skip
1283     \skip_if_eq:nnTF { \l__enumext_topsep_viii_skip } { \c_zero_skip }
1284       {
1285         \skip_set:Nn \l__enumext_minipage_left_skip
1286           {
1287             0.5\box_dp:N \strutbox
1288           }
1289         \skip_set:Nn \l__enumext_minipage_right_skip
1290           {
1291             \l__enumext_partopsep_viii_skip
1292           }
1293         \skip_set:Nn \l__enumext_minipage_after_skip
1294           {
1295             1.6\box_dp:N \strutbox
1296           }
1297       }
1298       {
```

```
1299        \skip_set:Nn \l__enumext_minipage_left_skip
1300          {
1301            0.5875\box_dp:N \strutbox
1302          }
1303        \skip_set:Nn \l__enumext_minipage_right_skip
1304          {
1305            \l__enumext_topsep_viii_skip
1306          }
1307        \skip_set:Nn \l__enumext_minipage_after_skip
1308          {
1309            0.325\box_dp:N \strutbox + \l__enumext_topsep_viii_skip
1310          }
1311      }
1312    }
```

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

__enumext_mini_addvspace_vii: The functions __enumext_mini_addvspace_vii: and __enumext_mini_addvspace_viii: will
__enumext_mini_addvspace_viii: apply the vertical space *"only above"* the __enumext_mini_env* environment on the *left side* when the
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.

```
1313  \cs_new_protected:Nn \__enumext_mini_addvspace_vii:
1314    {
1315      \__enumext_mini_set_vskip_vii:
1316      \par\nopagebreak
1317      \addvspace { \l__enumext_minipage_left_skip }
1318    }
1319  \cs_new_protected:Nn \__enumext_mini_addvspace_viii:
1320    {
1321      \__enumext_mini_set_vskip_viii:
1322      \par\nopagebreak
1323      \addvspace { \l__enumext_minipage_left_skip }
1324    }
```

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

11.20.4   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.

```
1325  \NewDocumentCommand \miniright { s }
1326    {
1327      \int_compare:nNnT { \l__enumext_keyans_pic_level_int } = { 1 }
1328        {
1329          \msg_error:nnn { enumext } { wrong-miniright-place }
1330        }
1331      \int_compare:nNnT { \l__enumext_level_int } = { 0 }
1332        {
1333          \msg_error:nnn { enumext } { wrong-miniright-place }
1334        }
1335      \int_compare:nNnTF { \l__enumext_keyans_level_int } = { 1 }
1336        {
1337          \__enumext_keyans_mini_right_cmd:n {#1}
1338        }
1339        { \__enumext_mini_right_cmd:n {#1} }
1340    }
```

(*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.

```
1341  \cs_new_protected:Npn \__enumext_mini_right_cmd:n #1
1342    {
1343      \dim_compare:nNnTF
1344        { \dim_use:c { l__enumext_minipage_right_ \__enumext_level: _dim } } > { \c_zero_dim }
1345        {
1346          \__enumext_multicols_stop:
1347          \end{__enumext_mini_env*}
1348          \hfill
1349          \begin{__enumext_mini_env*}
1350            { \dim_use:c { l__enumext_minipage_right_ \__enumext_level: _dim } }
1351            \par\addvspace { \l__enumext_minipage_right_skip }
1352            \bool_if:nF {#1}
1353              {
1354                \centering
1355              }
1356            \int_gzero:N \g__enumext_minipage_stat_int
1357          }
1358        { \msg_error:nnn { enumext } { wrong-miniright-use } }
1359    }
```

(*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.

```
1360  \cs_new_protected:Npn \__enumext_keyans_mini_right_cmd:n #1
1361    {
1362      \dim_compare:nNnTF { \l__enumext_minipage_right_v_dim } > { \c_zero_dim }
1363        {
1364          \__enumext_keyans_multicols_stop:
1365          \end{__enumext_mini_env*}
1366          \hfill
1367          \begin{__enumext_mini_env*}{ \l__enumext_minipage_right_v_dim }
1368            \par\addvspace { \l__enumext_minipage_right_skip }
1369            \bool_if:nF {#1}
1370              {
1371                \centering
1372              }
1373            \int_gzero:N \g__enumext_minipage_stat_int
1374          }
1375        { \msg_error:nnn { enumext } { wrong-miniright-use } }
1376    }
```

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

11.21   Setting above and below keys

While having controlled the *vertical spaces* within the `enumext` and `keyans` environments when using the `columns` or `mini-env` keys, sometimes the *"vertical spaces above"* or *"vertical spaces below"* the environments are not as expected and it is necessary to be able to apply a *"fine correction"* to these. As I have not been able to correct these *glitches*, the best option is to leave a couple of ⟨*keys*⟩ dedicated to this purpose, in this case it is best to use `\vspace` or `\vspace*` when convenient.

above
above*
below
below*
Define `above`, `above*`, `below` and `below*` keys for `enumext` and `keyans` environments.

```
1377  \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1378    {
1379      \keys_define:nn { enumext / #1 }
1380        {
1381          above  .skip_set:c = { l__enumext_vspace_above_#2_skip },
1382          above  .value_required:n = true,
1383          above* .code:n     = \bool_set_true:c { l__enumext_vspace_a_star_#2_bool }
1384                               \keys_set:nn { enumext / #1 } { above = {##1} },
1385          above* .value_required:n = true,
1386          below  .skip_set:c = { l__enumext_vspace_below_#2_skip },
1387          below  .value_required:n = true,
1388          below* .code:n     = \bool_set_true:c { l__enumext_vspace_b_star_#2_bool }
1389                               \keys_set:nn { enumext / #1 } { below = {##1} },
1390          below* .value_required:n = true,
```

```
1391            }
1392        }
1393    \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }
```

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

11.21.1   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.

```
1394    \cs_new_protected:Nn \__enumext_vspace_above:
1395        {
1396            \skip_if_eq:nnF
1397                { \skip_use:c { l__enumext_vspace_above_ \__enumext_level: _skip } } { \c_zero_skip }
1398                {
1399                    \bool_if:cTF { l__enumext_vspace_a_star_ \__enumext_level: _bool }
1400                        {
1401                            \vspace*{ \skip_use:c { l__enumext_vspace_above_ \__enumext_level: _skip } }
1402                        }
1403                        {
1404                            \vspace { \skip_use:c { l__enumext_vspace_above_ \__enumext_level: _skip } }
1405                        }
1406                }
1407        }
```

(*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.

```
1408    \cs_new_protected:Nn \__enumext_vspace_below:
1409        {
1410            \skip_if_eq:nnF
1411                { \skip_use:c { l__enumext_vspace_below_ \__enumext_level: _skip } } { \c_zero_skip }
1412                {
1413                    \bool_if:cTF { l__enumext_vspace_b_star_ \__enumext_level: _bool }
1414                        {
1415                            \vspace*{ \skip_use:c { l__enumext_vspace_below_ \__enumext_level: _skip } }
1416                        }
1417                        {
1418                            \vspace { \skip_use:c { l__enumext_vspace_below_ \__enumext_level: _skip } }
1419                        }
1420                }
1421        }
```

(*End of definition for __enumext_vspace_below:.*)

11.21.2   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.

```
1422    \cs_new_protected:Nn \__enumext_vspace_above_v:
1423        {
1424            \skip_if_eq:nnF { \l__enumext_vspace_above_v_skip } { \c_zero_skip }
1425                {
1426                    \bool_if:NTF \l__enumext_vspace_a_star_v_bool
1427                        {
1428                            \vspace*{ \l__enumext_vspace_above_v_skip }
1429                        }
1430                        { \vspace { \l__enumext_vspace_above_v_skip } }
1431                }
1432        }
```

(*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.

```
1433    \cs_new_protected:Nn \__enumext_vspace_below_v:
1434        {
1435            \skip_if_eq:nnF { \l__enumext_vspace_below_v_skip } { \c_zero_skip }
1436                {
1437                    \bool_if:NTF \l__enumext_vspace_b_star_v_bool
1438                        {
1439                            \vspace*{ \l__enumext_vspace_below_v_skip }
```

```
1440            }
1441         { \vspace { \l__enumext_vspace_below_v_skip } }
1442      }
1443   }
```

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

11.21.3   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.

```
1444 \cs_new_protected:Nn \__enumext_vspace_above_vii:
1445   {
1446      \skip_if_eq:nnF { \l__enumext_vspace_above_vii_skip } { \c_zero_skip }
1447         {
1448            \bool_if:NTF \l__enumext_vspace_a_star_vii_bool
1449               {
1450                  \vspace*{ \l__enumext_vspace_above_vii_skip }
1451               }
1452            { \vspace { \l__enumext_vspace_above_vii_skip } }
1453         }
1454   }
1455 \cs_new_protected:Nn \__enumext_vspace_above_viii:
1456   {
1457      \skip_if_eq:nnF { \l__enumext_vspace_above_viii_skip } { \c_zero_skip }
1458         {
1459            \bool_if:NTF \l__enumext_vspace_a_star_viii_bool
1460               {
1461                  \vspace*{ \l__enumext_vspace_above_viii_skip }
1462               }
1463            { \vspace { \l__enumext_vspace_above_viii_skip } }
1464         }
1465   }
```

(*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.

```
1466 \cs_new_protected:Nn \__enumext_vspace_below_vii:
1467   {
1468      \skip_if_eq:nnF { \l__enumext_vspace_below_vii_skip } { \c_zero_skip }
1469         {
1470            \bool_if:NTF \l__enumext_vspace_b_star_vii_bool
1471               {
1472                  \vspace*{ \l__enumext_vspace_below_vii_skip }
1473               }
1474            { \vspace { \l__enumext_vspace_below_vii_skip } }
1475         }
1476   }
1477 \cs_new_protected:Nn \__enumext_vspace_below_viii:
1478   {
1479      \skip_if_eq:nnF { \l__enumext_vspace_below_viii_skip } { \c_zero_skip }
1480         {
1481            \bool_if:NTF \l__enumext_vspace_b_star_viii_bool
1482               {
1483                  \vspace*{ \l__enumext_vspace_below_viii_skip }
1484               }
1485            { \vspace { \l__enumext_vspace_below_viii_skip } }
1486         }
1487   }
```

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

11.22   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*`.

```
1488 \cs_set_protected:Npn \__enumext_tmp:n #1
1489   {
1490     \keys_define:nn { enumext / #1 }
1491       {
1492         series  .str_set:N = \l__enumext_series_str,
1493         series  .value_required:n = true,
1494         resume  .code:n = \__enumext_resume_series:n {##1},
1495         resume* .code:n = \__enumext_resume_starred:,
1496         resume* .value_forbidden:n = true,
1497       }
1498   }
1499 \clist_map_inline:nn { level-1, enumext* } { \__enumext_tmp:n {#1} }
```

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

11.22.1   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.

```
1500 \cs_new:Npn \__enumext_filter_series:n #1
1501   {
1502     \use:e
1503       {
1504         \keyval_parse:NNn
1505           \__enumext_filter_series_key:n
1506           \__enumext_filter_series_pair:nn {#1}
1507       }
1508   }
```

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

```
1509 \cs_new:Npn \__enumext_filter_series_key:n #1
1510   {
1511     \str_case:nnF {#1}
1512       {
1513         { resume } {}
1514         { resume* } {}
1515       }
1516       { , { \exp_not:n {#1} } }
1517   }
```

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.

```
1518 \cs_new:Npn \__enumext_filter_series_pair:nn #1#2
1519   {
1520     \str_case:nnF {#1}
1521       {
1522         { series } {}
1523         { resume } {}
1524         { start } {}
1525         { save-ans } {}
1526         { save-key } {}
1527       }
1528       { , { \exp_not:n {#1} } = { \exp_not:n {#2} } }
1529   }
```

(*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 (§11.34) and to the function `\__enumext_parse_keys_vii:n` in the `enumext*` environment definition (§11.37).

```
1530 \cs_new_protected:Npn \__enumext_parse_series:n #1
1531   {
```

```
1532      \str_if_empty:NTF \l__enumext_series_str
1533        {
1534          \bool_if:NF \l__enumext_resume_active_bool
1535            {
1536              \__enumext_resume_last:n {#1}
1537            }
1538        }
1539        {
1540          \tl_gclear_new:c { g__enumext_series_ \l__enumext_series_str _tl }
1541          \tl_gset:ce { g__enumext_series_ \l__enumext_series_str _tl }
1542            { \__enumext_filter_series:n {#1} }
1543          \int_if_exist:cF { g__enumext_series_ \l__enumext_series_str _int }
1544            {
1545              \int_new:c { g__enumext_series_ \l__enumext_series_str _int }
1546            }
1547        }
1548    }
```

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.

```
1549  \cs_new_protected:Npn \__enumext_resume_last:n #1
1550    {
1551      \bool_if:NT \l__enumext_standar_first_bool
1552        {
1553          \tl_gclear:N \g__enumext_standar_series_tl
1554          \tl_gset:Ne \g__enumext_standar_series_tl { \__enumext_filter_series:n {#1} }
1555        }
1556      \bool_if:NT \l__enumext_starred_first_bool
1557        {
1558          \tl_gclear:N \g__enumext_starred_series_tl
1559          \tl_gset:Ne \g__enumext_starred_series_tl { \__enumext_filter_series:n {#1} }
1560        }
1561    }
```

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

11.22.2   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 (§11.34) and the enumext* environment definition (§11.37).

```
1562  \cs_new_protected:Nn \__enumext_resume_save_counter:
1563    {
1564      \bool_if:NT \g__enumext_standar_bool
1565        {
1566          \tl_if_empty:NF \l__enumext_series_str
1567            {
1568              \int_gset_eq:cN
1569                { g__enumext_series_ \l__enumext_series_str _int } \value{enumXi}
1570            }
1571          \tl_if_empty:NTF \l__enumext_resume_name_tl
1572            {
1573              \str_if_empty:NT \l__enumext_series_str
1574                {
1575                  \int_gset_eq:NN \g__enumext_resume_int \value{enumXi}
1576                }
1577            }
1578            {
1579              \int_if_exist:cT { g__enumext_series_ \l__enumext_resume_name_tl _int }
1580                {
1581                  \int_gset_eq:cN
```

```
1582                          { g__enumext_series_ \l__enumext_resume_name_tl _int } \value{enumXi}
1583                      }
1584                  }
1585              \int_if_exist:cT { g__enumext_resume_ \l__enumext_store_name_tl _int }
1586                  {
1587                      \int_gset_eq:cN
1588                          { g__enumext_resume_ \l__enumext_store_name_tl _int } \value{enumXi}
1589                  }
1590          }
1591      \bool_if:NT \g__enumext_starred_bool
1592          {
1593              \tl_if_empty:NF \l__enumext_series_str
1594                  {
1595                      \int_gset_eq:cN
1596                          { g__enumext_series_ \l__enumext_series_str _int } \value{enumXvii}
1597                  }
1598              \tl_if_empty:NTF \l__enumext_resume_name_tl
1599                  {
1600                      \str_if_empty:NT \l__enumext_series_str
1601                          {
1602                              \int_gset_eq:NN \g__enumext_resume_vii_int \value{enumXvii}
1603                          }
1604                  }
1605                  {
1606                      \int_if_exist:cT { g__enumext_series_ \l__enumext_resume_name_tl _int }
1607                          {
1608                              \int_gset_eq:cN
1609                                  { g__enumext_series_ \l__enumext_resume_name_tl _int } \value{enumXvii}
1610                          }
1611                  }
1612              \int_if_exist:cT { g__enumext_resume_ \l__enumext_store_name_tl _int }
1613                  {
1614                      \int_gset_eq:cN
1615                          { g__enumext_resume_ \l__enumext_store_name_tl _int } \value{enumXvii}
1616                  }
1617          }
1618      }
```

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

11.22.3 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.

```
1619 \cs_new_protected:Npn \__enumext_resume_series:n #1
1620      {
1621          \tl_if_empty:nTF {#1}
1622              {
1623                  \__enumext_resume_counter:n { }
1624              }
1625              {
1626                  \tl_if_exist:cTF { g__enumext_series_ \tl_to_str:n {#1} _tl }
1627                      {
1628                          \__enumext_resume_counter:n {#1}
1629                          \bool_if:NT \g__enumext_standar_bool
1630                              {
1631                                  \keys_set:nv { enumext / level-1 }
1632                                      { g__enumext_series_ \tl_to_str:n {#1} _tl }
1633                              }
1634                          \bool_if:NT \g__enumext_starred_bool
1635                              {
1636                                  \keys_set:nv { enumext / enumext* }
1637                                      { g__enumext_series_ \tl_to_str:n {#1} _tl }
1638                              }
1639                      }
1640                      {
1641                          \bool_if:NT \g__enumext_standar_bool
```

```
1642                    {
1643                        \msg_error:nnn { enumext } { unknown-series } {#1}
1644                    }
1645                \bool_if:NT \g__enumext_starred_bool
1646                    {
1647                        \msg_error:nnn { enumext } { unknown-series } {#1}
1648                    }
1649            }
1650        }
1651    }
```

(*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.

```
1652  \cs_new_protected:Npn \__enumext_resume_counter:n #1
1653    {
1654        \bool_set_true:N \l__enumext_resume_active_bool
1655        \tl_set:Nn \l__enumext_resume_name_tl {#1}
1656        \tl_if_empty:NTF \l__enumext_resume_name_tl
1657            {
1658                \__enumext_resume_counter:
1659            }
1660            {
1661                \__enumext_resume_counter_series:
1662            }
1663        \__enumext_resume_counter_save_ans:
1664    }
```

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.

```
1665  \cs_new_protected:Nn \__enumext_resume_counter:
1666    {
1667        \bool_if:NT \g__enumext_standar_bool
1668            {
1669                \int_gincr:N \g__enumext_resume_int
1670                \int_set_eq:NN \l__enumext_start_i_int \g__enumext_resume_int
1671            }
1672        \bool_if:NT \g__enumext_starred_bool
1673            {
1674                \int_gincr:N \g__enumext_resume_vii_int
1675                \int_set_eq:NN \l__enumext_start_vii_int \g__enumext_resume_vii_int
1676            }
1677    }
```

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.

```
1678  \cs_new_protected:Nn \__enumext_resume_counter_series:
1679    {
1680        \bool_if:NT \g__enumext_standar_bool
1681            {
1682                \int_set:Nn \l__enumext_start_i_int
1683                    {
1684                        \int_use:c { g__enumext_series_ \l__enumext_resume_name_tl _int } + 1
1685                    }
1686            }
1687        \bool_if:NT \g__enumext_starred_bool
1688            {
1689                \int_set:Nn \l__enumext_start_vii_int
1690                    {
1691                        \int_use:c { g__enumext_series_ \l__enumext_resume_name_tl _int } + 1
1692                    }
1693            }
1694    }
```

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.

```
1695 \cs_new_protected:Nn \__enumext_resume_counter_save_ans:
1696   {
1697     \bool_lazy_and:nnT
1698       { \bool_if_p:N \l__enumext_standar_first_bool }
1699       { \bool_if_p:N \l__enumext_store_active_bool }
1700       {
1701         \int_set:Nn \l__enumext_start_i_int
1702           {
1703             \int_use:c { g__enumext_resume_ \l__enumext_store_name_tl _int } + 1
1704           }
1705       }
1706     \bool_lazy_and:nnT
1707       { \bool_if_p:N \l__enumext_starred_first_bool }
1708       { \bool_if_p:N \l__enumext_store_active_bool }
1709       {
1710         \int_set:Nn \l__enumext_start_vii_int
1711           {
1712             \int_use:c { g__enumext_resume_ \l__enumext_store_name_tl _int } + 1
1713           }
1714       }
1715   }
```

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

11.22.4 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.

```
1716 \cs_new_protected:Nn \__enumext_resume_starred:
1717   {
1718     \bool_if:NT \g__enumext_standar_bool
1719       {
1720         \tl_if_empty:NF \g__enumext_standar_series_tl
1721           {
1722             \__enumext_resume_counter:n { }
1723             \keys_set:nV { enumext / level-1 } \g__enumext_standar_series_tl
1724           }
1725       }
1726     \bool_if:NT \g__enumext_starred_bool
1727       {
1728         \tl_if_empty:NF \g__enumext_starred_series_tl
1729           {
1730             \__enumext_resume_counter:n { }
1731             \keys_set:nV { enumext / enumext* } \g__enumext_starred_series_tl
1732           }
1733       }
1734   }
```

(*End of definition for* __enumext_resume_starred:.)

11.23 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.

11.23.1 Setting save-ans key

save-ans

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

```
1735 \cs_set_protected:Npn \__enumext_tmp:n #1
1736   {
1737     \keys_define:nn { enumext / #1 }
1738       {
1739         save-ans .code:n = \__enumext_storing_set:n {##1},
1740         save-ans .value_required:n = true,
1741       }
1742   }
1743 \clist_map_inline:nn { level-1, enumext* } { \__enumext_tmp:n {#1} }
```

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

11.23.2 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:.

```
1744 \cs_new_protected:Nn \__enumext_start_save_ans_msg:
1745   {
1746     \msg_term:nnVV { enumext } { save-ans-log }
1747       \g__enumext_envir_name_tl \l__enumext_store_name_tl
1748   }
1749 \cs_new_protected:Nn \__enumext_stop_save_ans_msg:
1750   {
1751     \msg_term:nnVV { enumext } { save-ans-log-hook }
1752       \g__enumext_envir_name_tl \g__enumext_store_name_tl
1753   }
```

(*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.

```
1754 \cs_new_protected:Npn \__enumext_storing_set:n #1
1755   {
1756     \tl_set:Ne \l__enumext_store_name_tl {#1}
1757     \tl_if_empty:NTF \l__enumext_store_name_tl
1758       {
1759         \bool_lazy_or:nnT
1760           { \l__enumext_standar_first_bool } { \l__enumext_starred_first_bool }
1761           {
1762             \msg_error:nnV { enumext } { save-ans-empty } \g__enumext_envir_name_tl
1763           }
1764       }
1765       {
1766         \bool_lazy_or:nnT
1767           { \l__enumext_standar_first_bool } { \l__enumext_starred_first_bool }
1768           {
1769             \__enumext_start_save_ans_msg:
1770             \__enumext_storing_exec:
1771           }
1772       }
1773   }
```

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 anskeyenv (§11.27). 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*.

```
1774 \cs_new_protected:Nn \__enumext_storing_exec:
1775   {
1776     \bool_set_true:N \l__enumext_store_active_bool
1777     \bool_set_true:N \l__enumext_check_answers_bool
1778     \tl_gset:NV \g__enumext_store_name_tl \l__enumext_store_name_tl
1779     \__enumext_anskey_env_make:V \l__enumext_store_name_tl
1780     \prop_if_exist:cF { g__enumext_ \l__enumext_store_name_tl _prop }
1781       {
1782         \msg_log:nnV { enumext } { store-prop } \l__enumext_store_name_tl
1783         \prop_new:c { g__enumext_ \l__enumext_store_name_tl _prop }
1784       }
1785     \seq_if_exist:cF { g__enumext_ \l__enumext_store_name_tl _seq }
1786       {
1787         \msg_log:nnV { enumext } { store-seq } \l__enumext_store_name_tl
1788         \seq_new:c { g__enumext_ \l__enumext_store_name_tl _seq }
```

```
1789              }
1790       \int_if_exist:cF { g__enumext_resume_ \l__enumext_store_name_tl _int }
1791         {
1792             \msg_log:nnV { enumext } { store-int } \l__enumext_store_name_tl
1793             \int_new:c { g__enumext_resume_ \l__enumext_store_name_tl _int }
1794         }
1795     }
```

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

11.23.3 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.

11.23.4 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.

```
1796 \cs_set_protected:Npn \__enumext_tmp:n #1
1797   {
1798     \keys_define:nn { enumext / #1 }
1799       {
1800         check-ans .bool_set:N = \l__enumext_check_ans_key_bool,
1801         check-ans .initial:n  = false,
1802         check-ans .value_required:n = true,
1803         no-store  .code:n = {
1804                             \bool_set_false:N \l__enumext_check_answers_bool
1805                             \bool_set_false:N \l__enumext_check_ans_key_bool
1806                           },
1807         no-store  .value_forbidden:n = true,
1808       }
1809   }
1810 \clist_map_inline:nn
1811   {
1812     level-1, level-2, level-3, level-4, enumext*
1813   }
1814   { \__enumext_tmp:n {#1} }
```

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

11.23.5 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.

```
1815 \cs_new_protected:Nn \__enumext_check_ans_active:
1816   {
1817     \tl_if_empty:NF \l__enumext_store_name_tl
1818       {
1819         \bool_if:NT \l__enumext_check_answers_bool
1820           {
1821             \__enumext_check_ans_level:
1822           }
1823       }
1824   }
```

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.

```
1825 \cs_new_protected:Nn \__enumext_check_ans_level:
1826   {
1827     \int_case:nn { \l__enumext_level_int }
1828       {
1829         { 1 }{
1830             \bool_lazy_all:nT
1831               {
1832                 { \bool_if_p:N \g__enumext_starred_bool }
1833                 { \int_compare_p:nNn { \l__enumext_level_h_int } = { 1 } }
1834               }
1835               {
1836                 \int_gdecr:N \g__enumext_item_number_int
1837               }
1838         }
1839         { 2 }{
1840             \int_gdecr:N \g__enumext_item_number_int
1841         }
1842         { 3 }{
1843             \int_gdecr:N \g__enumext_item_number_int
1844         }
1845         { 4 }{
1846             \int_gdecr:N \g__enumext_item_number_int
1847         }
1848       }
```

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.

```
1849     \int_case:nn { \l__enumext_level_h_int }
1850       {
1851         { 1 }{
1852             \bool_lazy_all:nT
1853               {
1854                 { \bool_if_p:N \g__enumext_standar_bool }
1855                 { \int_compare_p:nNn { \l__enumext_level_int } = { 1 } }
1856               }
1857               {
1858                 \int_gdecr:N \g__enumext_item_number_int
1859               }
1860         }
1861       }
1862   }
```

(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.

```
1863 \cs_new_protected:Nn \__enumext_check_ans_key_hook:
1864   {
1865     \bool_lazy_and:nnT
```

```
1866        { \bool_if_p:N \l__enumext_check_ans_key_bool }
1867        { \bool_if_p:N \g__enumext_standar_bool }
1868        {
1869            \bool_gset_true:N \g__enumext_check_ans_key_bool
1870        }
1871      \bool_lazy_and:nnT
1872        { \bool_if_p:N \l__enumext_check_ans_key_bool }
1873        { \bool_if_p:N \g__enumext_starred_bool }
1874        {
1875            \bool_gset_true:N \g__enumext_check_ans_key_bool
1876        }
1877    }
```

(*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:.

```
1878  \cs_new_protected:Nn \__enumext_item_answer_diff:
1879    {
1880      \int_gset:Nn \g__enumext_item_answer_diff_int
1881        {
1882          \int_sign:n { \g__enumext_item_number_int - \g__enumext_item_anskey_int }
1883        }
1884    }
```

(*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_-
__enumext_check_ans_msg_less:　execute_after_env: when the key check-ans is active, that is, when \g__enumext_check_ans_-
__enumext_check_ans_msg_same_ok:　key_bool is *"true"* and will return the appropriate message according to the value of \g__enumext_-
__enumext_check_ans_msg_greater:　item_answer_diff_int set by the function __enumext_item_answer_diff:.

```
1885  \cs_new_protected:Nn \__enumext_check_ans_show:
1886    {
1887      \int_case:nn { \g__enumext_item_answer_diff_int }
1888        {
1889          { -1 }{ \__enumext_check_ans_msg_less:    }
1890          {  0 }{ \__enumext_check_ans_msg_same_ok: }
1891          {  1 }{ \__enumext_check_ans_msg_greater: }
1892        }
1893    }
1894  \cs_new_protected:Nn \__enumext_check_ans_msg_less:
1895    {
1896      \msg_warning:nneee { enumext } { item-less-answer } { \g__enumext_store_name_tl }
1897        { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
1898    }
1899  \cs_new_protected:Nn \__enumext_check_ans_msg_same_ok:
1900    {
1901      \msg_term:nneee { enumext } { items-same-answer } { \g__enumext_store_name_tl }
1902        { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
1903    }
1904  \cs_new_protected:Nn \__enumext_check_ans_msg_greater:
1905    {
1906      \msg_warning:nneee { enumext } { item-greater-answer } { \g__enumext_store_name_tl }
1907        { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
1908    }
```

(*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_-
__enumext_check_ans_log_msg_less:　execute_after_env: when the key check-ans is not active, that is, when \g__enumext_check_-
__enumext_check_ans_log_msg_same_ok:　ans_key_bool is *"false"* and write in the log the appropriate message according to the value of \g__-
__enumext_check_ans_log_msg_greater:　enumext_item_answer_diff_int set by the function __enumext_item_answer_diff:.

```
1909  \cs_new_protected:Nn \__enumext_check_ans_log:
1910    {
1911      \int_case:nn { \g__enumext_item_answer_diff_int }
1912        {
1913          { -1 }{ \__enumext_check_ans_log_msg_less:    }
1914          {  0 }{ \__enumext_check_ans_log_msg_same_ok: }
```

```
1915            {  1 }{ \__enumext_check_ans_log_msg_greater: }
1916          }
1917    }
1918 \cs_new_protected:Nn \__enumext_check_ans_log_msg_less:
1919    {
1920      \msg_log:nneee { enumext } { item-less-answer } { \g__enumext_store_name_tl }
1921        { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
1922    }
1923 \cs_new_protected:Nn \__enumext_check_ans_log_msg_same_ok:
1924    {
1925      \msg_log:nneee { enumext } { items-same-answer }  { \g__enumext_store_name_tl }
1926        { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
1927    }
1928 \cs_new_protected:Nn \__enumext_check_ans_log_msg_greater:
1929    {
1930      \msg_log:nneee { enumext } { item-greater-answer } { \g__enumext_store_name_tl }
1931        { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
1932    }
```

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

11.23.6 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.

```
1933 \cs_new_protected:Npn \__enumext_check_starred_cmd:n #1
1934    {
1935      \int_compare:nNnT
1936        { \g__enumext_check_starred_cmd_int } = { 0 }
1937        {
1938          \msg_warning:nnnV
1939            { enumext } { missing-starred }{ #1 } \l__enumext_check_start_line_env_tl
1940        }
1941      \int_compare:nNnT
1942        { \g__enumext_check_starred_cmd_int } > { 1 }
1943        {
1944          \msg_warning:nnnV
1945            { enumext } { many-starred }{ #1 } \l__enumext_check_start_line_env_tl
1946        }
1947      \int_gzero:N \g__enumext_check_starred_cmd_int
1948      \tl_clear:N \l__enumext_check_start_line_env_tl
1949    }
```

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

11.24 Writing .log, executing check-ans key and more

__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 anskeyenv (§11.27) 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.

```
1950 \cs_new_protected:Nn \__enumext_execute_after_env:
1951    {
1952      \int_compare:nNnT { \l__enumext_level_int } = { 0 }
1953        {
1954          \tl_if_empty:NF \g__enumext_store_name_tl
1955            {
1956              \__enumext_stop_save_ans_msg:
1957              \__enumext_item_answer_diff:
1958              \__enumext_log_global_vars:
1959              \__enumext_log_answer_vars:
1960              \bool_if:NTF \g__enumext_check_ans_key_bool
1961                {
1962                  \__enumext_check_ans_show:
1963                }
1964                { \__enumext_check_ans_log: }
1965              \__enumext_undefine_anskey_env:
```

```
1966                 }
1967             \__enumext_reset_global_vars:
1968         }
1969     }
```

💣 This function is passed to the function __enumext_after_env:nn for the environments enumext (§11.34) and
enumext* (§11.37) 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:.)

11.25   Keys and functions associated with storage

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

mark-ans
mark-pos
show-ans
mark-ref
save-ref

```
1970 \cs_set_protected:Npn \__enumext_tmp:n #1
1971   {
1972     \keys_define:nn { enumext / #1 }
1973       {
1974         wrap-ans   .cs_set_protected:Np = \__enumext_anskey_wrapper:n ##1,
1975         wrap-ans   .initial:n = \fbox{##1},
1976         wrap-ans   .value_required:n = true,
1977         wrap-opt   .cs_set_protected:Np = \__enumext_keyans_wrapper_opt:n ##1,
1978         wrap-opt   .initial:n = [{##1}],
1979         wrap-opt   .value_required:n = true,
1980         save-sep   .tl_set:N  = \l__enumext_store_keyans_item_opt_sep_tl,
1981         save-sep   .initial:n = {, ~ },
1982         save-sep   .value_required:n = true,
1983         mark-ans   .tl_set:N  = \l__enumext_mark_answer_sym_tl,
1984         mark-ans   .initial:n = \textasteriskcentered,
1985         mark-ans   .value_required:n = true,
1986         mark-pos   .choice:,
1987         mark-pos / left    .code:n = \str_set:Nn \l__enumext_mark_position_str { l },
1988         mark-pos / right   .code:n = \str_set:Nn \l__enumext_mark_position_str { r },
1989 %%      mark-pos / unknown .code:n =
1990 %%                          \msg_error:nneee { enumext } { unknown-choice }
1991 %%                            { mark-pos } { left , right } { \exp_not:n {##1} },
1992         mark-pos   .initial:n = right,
1993         mark-pos   .value_required:n = true,
1994         show-ans   .bool_set:N = \l__enumext_show_answer_bool,
1995         show-ans   .initial:n  = false,
1996         show-ans   .value_required:n = true,
1997         show-pos   .bool_set:N = \l__enumext_show_position_bool,
1998         show-pos   .initial:n  = false,
1999         show-pos   .value_required:n = true,
2000         mark-ref   .tl_set:N   = \l__enumext_mark_ref_sym_tl,
2001         mark-ref   .initial:n  = \textasteriskcentered,
2002         mark-ref   .value_required:n = true,
2003         save-ref   .bool_set:N = \l__enumext_store_ref_key_bool,
2004         save-ref   .initial:n  = false,
2005         save-ref   .value_required:n = true,
2006       }
2007   }
2008 \clist_map_inline:nn { level-1, enumext* } { \__enumext_tmp:n {#1} }
```

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

mark-pos For the keyans and keyans* environments we will only add the keys mark-pos, show-ans and show-
show-ans pos.
show-pos
```
2009 \cs_set_protected:Npn \__enumext_tmp:n #1
2010   {
2011     \keys_define:nn { enumext / #1 }
2012       {
2013         mark-pos .choice:,
2014         mark-pos / left   .code:n = \str_set:Nn \l__enumext_mark_position_str { l },
2015         mark-pos / right  .code:n = \str_set:Nn \l__enumext_mark_position_str { r },
2016         mark-pos .initial:n = right,
2017         mark-pos .value_required:n  = true,
2018         show-ans .bool_set:N = \l__enumext_show_answer_bool,
2019         show-ans .initial:n  = false,
2020         show-ans .value_required:n = true,
2021         show-pos .bool_set:N = \l__enumext_show_position_bool,
```

```
2022        show-pos .initial:n  = false,
2023        show-pos .value_required:n = true,
2024      }
2025   }
2026 \clist_map_inline:nn { keyans, keyans* } { \__enumext_tmp:n {#1} }
```

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

11.25.1 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 The functions __enumext_store_active_keys:n and __enumext_store_active_keys_vii:n
__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*⟩.

```
2027 \cs_new_protected:Npn \__enumext_store_active_keys:n #1
2028   {
2029     \bool_if:cF { l__enumext_store_save_key_ \__enumext_level: _bool }
2030       {
2031         \tl_clear:c { l__enumext_save_key_ \__enumext_level: _tl }
2032         \tl_set:ce
2033           { l__enumext_store_save_key_ \__enumext_level: _tl }
2034           { \__enumext_filter_save_key:n {#1} }
2035       }
2036   }
2037 \cs_new_protected:Npn \__enumext_store_active_keys_vii:n #1
2038   {
2039     \bool_if:NF \l__enumext_store_save_key_vii_bool
2040       {
2041         \tl_clear:N \l__enumext_store_save_key_vii_tl
2042         \tl_set:Ne \l__enumext_store_save_key_vii_tl { \__enumext_filter_save_key:n {#1} }
2043       }
2044   }
```

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

11.25.2 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.

```
2045 \cs_set_protected:Npn \__enumext_tmp:n #1
2046   {
2047     \keys_define:nn { enumext / enumext* }
2048       {
2049         save-key .code:n = \__enumext_parse_save_key_vii:n {##1},
2050         save-key .value_required:n = true,
2051       }
2052     \keys_define:nn { enumext / #1 }
2053       {
2054         save-key .code:n = \__enumext_parse_save_key:n {##1},
2055         save-key .value_required:n = true,
2056       }
2057   }
2058 \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 The functions __enumext_parse_save_key:n and __enumext_parse_save_key_vii:n will be
__enumext_parse_save_key_vii:n responsible for storing the filtered ⟨*keys*⟩ in the variable \l__enumext_store_save_key_X_tl for enumext and enumext*.

```
2059 \cs_new_protected:Npn \__enumext_parse_save_key:n #1
```

```
2060    {
2061      \bool_set_true:c { l__enumext_store_save_key_ \__enumext_level: _bool }
2062      \tl_clear:c { l__enumext_save_key_ \__enumext_level: _tl }
2063      \tl_set:ce
2064        { l__enumext_store_save_key_ \__enumext_level: _tl }
2065        { \__enumext_filter_save_key:n {#1} }
2066    }
2067 \cs_new_protected:Npn \__enumext_parse_save_key_vii:n #1
2068    {
2069      \bool_set_true:N \l__enumext_store_save_key_vii_bool
2070      \tl_clear:N \l__enumext_store_save_key_vii_tl
2071      \tl_set:Ne \l__enumext_store_save_key_vii_tl { \__enumext_filter_save_key:n {#1} }
2072    }
```

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

11.25.3 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.

```
2073 \cs_new:Npn \__enumext_filter_save_key:n #1
2074    {
2075      \use:e
2076        {
2077          \keyval_parse:NNn
2078            \__enumext_filter_save_key_key:n
2079            \__enumext_filter_save_key_pair:nn {#1}
2080        }
2081    }
```

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* and no-store keys.

```
2082 \cs_new:Npn \__enumext_filter_save_key_key:n #1
2083    {
2084      \str_case:nnF {#1}
2085        {
2086          { resume } {} { resume* } {} { no-store } {}
2087        }
2088        { , { \exp_not:n {#1} } }
2089    }
```

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.

```
2090 \cs_new:Npn \__enumext_filter_save_key_pair:nn #1#2
2091    {
2092      \str_case:nnF {#1}
2093        {
2094          { series   } {} { resume   } {} { save-ans } {}
2095          { save-ref } {} { save-key } {} { check-ans } {} { show-ans } {}
2096          { show-pos } {} { wrap-ans } {} { mark-ans  } {} { wrap-opt } {}
2097          { save-sep } {} { mark-ref } {} { mini-env  } {} { mini-sep } { }
2098          { mini-right } {} { mini-right* } {}
2099        }
2100        { , { \exp_not:n {#1} } = { \exp_not:n {#2} } }
2101    }
```

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

11.25.4 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.

```
2102 \cs_new_protected:Npn \__enumext_store_addto_prop:n #1
2103    {
2104      \prop_gput_if_not_in:cen { g__enumext_ \l__enumext_store_name_tl _prop }
2105        {
2106          \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } + 1 }
```

```
2107              }
2108            { #1 }
2109        }
2110    \cs_generate_variant:Nn \__enumext_store_addto_prop:n { V, e }
```

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

11.25.5 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.

```
2111    \cs_new_protected:Npn \__enumext_store_addto_seq:n #1
2112      {
2113         \seq_gput_right:cn { g__enumext_ \l__enumext_store_name_tl _seq } { #1 }
2114      }
2115    \cs_generate_variant:Nn \__enumext_store_addto_seq:n { v, V, e }
```

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

11.25.6 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.

```
2116    \cs_new_protected:Nn \__enumext_store_level_open:
2117      {
2118         \bool_if:NT \l__enumext_check_answers_bool
2119           {
2120              \tl_if_empty:cTF { l__enumext_store_save_key_ \__enumext_level: _tl }
2121                {
2122                   \__enumext_store_addto_seq:n
2123                     {
2124                        \item \begin{enumext}
2125                     }
2126                }
2127                {
2128                   \tl_put_left:cn { l__enumext_store_save_key_ \__enumext_level: _tl }
2129                     {
2130                        \item \begin{enumext} [
2131                     }
2132                   \tl_put_right:cn { l__enumext_store_save_key_ \__enumext_level: _tl }
2133                     {
2134                        ]
2135                     }
2136                   \__enumext_store_addto_seq:v { l__enumext_store_save_key_ \__enumext_level: _tl }
2137                }
2138           }
2139      }
2140    \cs_new_protected:Nn \__enumext_store_level_close:
2141      {
2142         \bool_if:NT \l__enumext_check_answers_bool
2143           {
2144              \__enumext_store_addto_seq:n { \end{enumext} }
2145           }
2146      }
```

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

__enumext_store_level_open_vii:
__enumext_store_level_close_vii:

When nesting the enumext* environment in enumext starting right after \item (without material between them) there is a problem with the alignment of the labels with the baseline between the two environments. One way to get around this problem is to place \mode_leave_vertical: and then apply \vspace taking into account \baselineskip, the value of \parsep of the current level of enumext and the value of \topsep of the enumext* environment.

```
2147    \cs_new_protected:Nn \__enumext_store_level_open_vii:
2148      {
2149         \bool_if:NT \l__enumext_check_answers_bool
2150           {
2151              \tl_if_empty:NTF \l__enumext_store_save_key_vii_tl
2152                {
2153                   \__enumext_store_addto_seq:n
```

```
2154              {
2155                \item \mode_leave_vertical:
2156                  \vspace { -\skip_eval:n { \baselineskip + \parsep } }
2157                  \begin{enumext*}[before={\setlength{\topsep}{0pt}},]
2158              }
2159            }
2160            {
2161              \tl_put_left:Nn \l__enumext_store_save_key_vii_tl
2162                {
2163                  \item \mode_leave_vertical:
2164                    \vspace { -\skip_eval:n { \baselineskip + \parsep } }
2165                    \begin{enumext*}[before={\setlength{\topsep}{0pt}},
2166                }
2167              \tl_put_right:Nn \l__enumext_store_save_key_vii_tl
2168                {
2169                  ]
2170                }
2171              \__enumext_store_addto_seq:V \l__enumext_store_save_key_vii_tl
2172            }
2173        }
2174      }
2175 \cs_new_protected:Nn \__enumext_store_level_close_vii:
2176    {
2177      \bool_if:NT \l__enumext_check_answers_bool
2178        {
2179          \__enumext_store_addto_seq:n { \end{enumext*} }
2180        }
2181    }
```

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

11.25.7 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

```
2182 \cs_new_protected:Nn \__enumext_print_keyans_box:NN
2183    {
2184      \mode_leave_vertical:
2185      \skip_horizontal:n { -\dim_use:N #2 }
2186      \makebox[0pt][ r ]
2187        {
2188          \makebox[ \dim_use:N #1 ][ \l__enumext_mark_position_str ]
2189            {
2190              \tl_use:N \l__enumext_mark_answer_sym_tl
2191            }
2192        }
2193      \skip_horizontal:n { \dim_use:N #2 }
2194    }
2195 \cs_generate_variant:Nn \__enumext_print_keyans_box:NN { cc }
```

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

11.26 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*.

```
2196 \keys_define:nn { enumext / anskey }
2197    {
2198      break-col .bool_set:N = \l__enumext_store_columns_break_bool,
2199      break-col .default:n  = true,
2200      break-col .value_forbidden:n = true,
2201      item-join .int_set:N = \l__enumext_store_item_join_int,
2202      item-join .value_required:n = true,
2203      item-star .bool_set:N = \l__enumext_store_item_star_bool,
2204      item-star .default:n  = true,
```

```
2205    item-star .value_forbidden:n = true,
2206    item-sym* .tl_set:N   = \l__enumext_store_item_symbol_tl,
2207    item-sym* .value_required:n = true,
2208    item-pos* .dim_set:N  = \l__enumext_store_item_symbol_sep_dim,
2209    item-pos* .value_required:n = true,
2210    }
```

💣 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 we call the function `\__enumext_store_anskey_-code:nn`.

```
2211  \NewDocumentCommand \anskey { o +m }
2212    {
2213      \__enumext_anskey_safe_outer:
2214      \group_begin:
2215        \bool_if:NT \l__enumext_check_answers_bool
2216          {
2217            \int_gincr:N \g__enumext_item_anskey_int
2218            \__enumext_anskey_safe_inner:n {#2}
2219            \__enumext_store_anskey_code:nn {#1} {#2}
2220          }
2221      \group_end:
2222    }
```

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

11.26.1   Internal functions for the command

`\__enumext_anskey_safe_outer:`
`\__enumext_anskey_safe_inner:n`

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.

```
2223  \cs_new_protected:Nn \__enumext_anskey_safe_outer:
2224    {
2225      \bool_if:NF \l__enumext_store_active_bool
2226        {
2227          \msg_error:nnnn { enumext } { anskey-wrong-place }{ anskey }{ enumext }
2228        }
2229      \int_compare:nNnT { \l__enumext_keyans_level_int } = { 1 }
2230        {
2231          \msg_error:nnnn { enumext } { command-wrong-place }{ anskey }{ keyans }
2232        }
2233      \int_compare:nNnT { \l__enumext_keyans_pic_level_int } = { 1 }
2234        {
2235          \msg_error:nnnn { enumext } { command-wrong-place }{ anskey }{ keyanspic }
2236        }
2237    }
```

The `\__enumext_anskey_safe_inner:n` function will first check to see if the passed argument is empty and then check to see if the command is nested by returning the appropriate messages.

```
2238  \cs_new_protected:Npn \__enumext_anskey_safe_inner:n #1
2239    {
2240      \tl_if_empty:nT {#1}
2241        {
2242          \msg_error:nn { enumext } { anskey-empty-arg }
2243        }
2244      \int_incr:N \l__enumext_anskey_level_int
2245      \int_compare:nNnT { \l__enumext_anskey_level_int } > { 1 }
2246        {
2247          \msg_error:nn { enumext } { anskey-nested }
2248        }
2249    }
```

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

`\__enumext_store_anskey_code:nn` The internal function `\__enumext_store_anskey_code:nn` 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⟩ passed to the command.

```
2250  \cs_new_protected:Npn \__enumext_store_anskey_code:nn #1 #2
2251    {
2252      \__enumext_store_addto_prop:n {#2}
2253      \bool_if:NT \l__enumext_store_ref_key_bool
2254        {
2255          \__enumext_store_internal_ref:
2256        }
2257      \__enumext_store_anskey_show_left:n { #2 }
```

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.

```
2258      \tl_if_novalue:nF {#1}
2259        {
2260          \keys_set:nn { enumext / anskey } {#1}
2261        }
2262      \tl_clear:N \l__enumext_store_anskey_arg_tl
2263      \bool_lazy_and:nnT
2264        { \bool_if_p:N \l__enumext_store_columns_break_bool }
2265        { \bool_not_p:n { \l__enumext_starred_bool } }
2266        {
2267          \tl_put_left:Nn \l__enumext_store_anskey_arg_tl { \columnbreak }
2268        }
2269      \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.

```
2270      \bool_lazy_and:nnT
2271        { \bool_not_p:n { \l__enumext_starred_bool } }
2272        { \int_compare_p:nNn { \l__enumext_store_item_join_int } > { 1 } }
2273        {
2274          \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2275            {
2276              ( \exp_not:V \l__enumext_store_item_join_int )
2277            }
2278        }
```

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⟩.

```
2279      \bool_if:NTF \l__enumext_store_item_star_bool
2280        {
2281          \tl_put_right:Nn \l__enumext_store_anskey_arg_tl { * }
2282          \tl_if_empty:NF \l__enumext_store_item_symbol_tl
2283            {
2284              \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2285                {
2286                  [ \exp_not:V \l__enumext_store_item_symbol_tl ]
2287                }
2288            }
2289          \dim_compare:nT
2290            {
2291              \l__enumext_store_item_symbol_sep_dim != \c_zero_dim
2292            }
2293            {
2294              \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2295                {
2296                  [ \exp_not:V \l__enumext_store_item_symbol_sep_dim ]
2297                }
2298            }
2299          \tl_put_right:Nn \l__enumext_store_anskey_arg_tl {#2}
2300        }
2301        {
2302          \tl_put_right:Nn \l__enumext_store_anskey_arg_tl {#2}
2303        }
```

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⟩.

```
2304      \bool_lazy_and:nnT
2305        { \bool_if_p:N \l__enumext_store_ref_key_bool }
```

```
2306          { \bool_if_p:N \l__enumext_hyperref_bool }
2307          {
2308            \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2309              {
2310                \hfill \exp_not:N \hyperlink { \exp_not:V \l__enumext_newlabel_arg_one_tl }
2311                  { \exp_not:V \l__enumext_mark_ref_sym_tl }
2312              }
2313          }
2314        \__enumext_store_addto_seq:V \l__enumext_store_anskey_arg_tl
2315    }
```

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

__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.

```
2316  \cs_new_protected:Nn \__enumext_store_internal_ref:
2317    {
2318      \cs_set_protected:Npn \__enumext_tmp:n ##1
2319        {
2320          \tl_set_eq:cc { l__enumext_label_copy_##1_tl } { l__enumext_label_##1_tl }
2321          \tl_reverse:c { l__enumext_label_copy_##1_tl }
2322          \tl_remove_once:cn { l__enumext_label_copy_##1_tl } { . }
2323          \tl_reverse:c { l__enumext_label_copy_##1_tl }
2324        }
2325      \clist_map_inline:nn { i, ii, iii, iv, vii } { \__enumext_tmp:n {##1} }
2326      \cs_set:Npn \__enumext_tmp:n ##1
2327        { . \tl_use:c { l__enumext_label_copy_ \int_to_roman:n {##1} _tl } }
```

Here we need to analyse the cases where the environment is started with enumext* and if \anskey is running alone in it or if it is running in a nested enumext environment within the starting environment.

```
2328        \bool_lazy_all:nT
2329          {
2330            { \bool_if_p:N \g__enumext_starred_bool }
2331            { \int_compare_p:nNn { \l__enumext_level_int } = { \c_zero_int } }
2332          }
2333          {
2334            \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2335              { \tl_use:N \l__enumext_label_copy_vii_tl }
2336          }
2337        \bool_lazy_all:nT
2338          {
2339            { \bool_if_p:N \l__enumext_standar_bool }
2340            { \bool_if_p:N \g__enumext_starred_bool }
2341            { \int_compare_p:nNn { \l__enumext_level_int } > { \c_zero_int } }
2342          }
2343          {
2344            \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2345              {
2346                \tl_use:N \l__enumext_label_copy_vii_tl
2347                \int_step_function:nnN { 1 } { \l__enumext_level_int } \__enumext_tmp:n
2348              }
2349          }
```

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

```
2350        \bool_lazy_all:nT
2351          {
2352            { \bool_if_p:N \l__enumext_standar_bool }
2353            { \int_compare_p:nNn { \l__enumext_level_int } > { \c_zero_int } }
2354            { \int_compare_p:nNn { \l__enumext_level_h_int } = { \c_zero_int } }
2355            { \bool_not_p:n { \l__enumext_starred_bool } }
2356          }
2357          {
2358            \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2359              {
2360                \tl_use:N \l__enumext_label_copy_i_tl
2361                \int_step_function:nnN { 2 } { \l__enumext_level_int } \__enumext_tmp:n
2362              }
```

```
2363           }
2364         \cs_set:Npn \__enumext_tmp:n ##1
2365           { \tl_use:c { l__enumext_label_copy_ \int_to_roman:n {##1} _tl } }
2366         \bool_lazy_all:nT
2367           {
2368             { \bool_if_p:N \l__enumext_standar_bool }
2369             { \int_compare_p:nNn { \l__enumext_level_int } > { \c_zero_int } }
2370             { \bool_not_p:n { \g__enumext_starred_bool } }
2371             { \int_compare_p:nNn { \l__enumext_level_h_int } > { \c_zero_int } }
2372           }
2373           {
2374             \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2375               {
2376                 \int_step_function:nnN { 1 } { \l__enumext_level_int } \__enumext_tmp:n
2377                 . \tl_use:N \l__enumext_label_copy_vii_tl
2378               }
2379           }
```

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

```
2380         \tl_put_right:Ne \l__enumext_newlabel_arg_one_tl
2381           {
2382             \l__enumext_store_name_tl \c_colon_str
2383             \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } }
2384           }
```

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.

```
2385         \tl_put_right:Ne \l__enumext_write_aux_file_tl
2386           {
2387             \__enumext_newlabel:nn
2388               { \exp_not:V \l__enumext_newlabel_arg_one_tl }
2389               { \l__enumext_newlabel_arg_two_tl }
2390           }
2391         \l__enumext_write_aux_file_tl
2392       }
```

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

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

```
2393     \cs_new_protected:Npn \__enumext_store_anskey_show_wrap:n #1
2394       {
2395         \par
2396         \bool_if:NT \l__enumext_starred_bool
2397           {
2398             \cs_set:Nn \__enumext_level: { vii }
2399           }
2400         \__enumext_print_keyans_box:cc
2401           { l__enumext_labelwidth_ \__enumext_level: _dim }
2402           { l__enumext_labelsep_ \__enumext_level: _dim }
2403         \__enumext_anskey_wrapper:n { #1 }
2404       }
```

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

__enumext_store_anskey_show_left:n   The function __enumext_store_anskey_show_left:n will show the "*mark*" defined by the mark-ans key or the "*position*" of the content stored in the ⟨*prop list*⟩ when using the show-pos key on the left margin next to the "*wraps*" ⟨*argument*⟩ passed to \anskey on the right side when using the show-ans key.

```
2405     \cs_new_protected:Npn \__enumext_store_anskey_show_left:n #1
2406       {
2407         \bool_if:NT \l__enumext_show_answer_bool
2408           {
2409             \__enumext_store_anskey_show_wrap:n { #1 }
2410           }
2411         \bool_if:NT \l__enumext_show_position_bool
2412           {
2413             \tl_set:Ne \l__enumext_mark_answer_sym_tl
2414               {
2415                 \group_begin:
```

```
2416              \exp_not:N \normalfont
2417              \exp_not:N \footnotesize [ \int_eval:n
2418                {
2419                  \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
2420                }
2421                ]
2422              \group_end:
2423            }
2424          \__enumext_store_anskey_show_wrap:n { #1 }
2425        }
2426    }
```

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

11.27   The environment anskeyenv

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 fatal error, then the user will have to see how to solve the problem.

__enumext_undefine_anskey_env: Detection of the anskeyenv environment outside the enumext and enumext* environments.

```
2427  \__enumext_before_env:nn { enumext }
2428    {
2429      \int_compare:nNnT { \l__enumext_level_int } = { 0 }
2430        {
2431          \cs_if_free:NF \__scontents_anskeyenv_env_begin:
2432            {
2433              \msg_fatal:nnn { enumext } { anskeyenv-defined } { anskeyenv }
2434            }
2435        }
2436    }
2437  \__enumext_before_env:nn { enumext* }
2438    {
2439      \int_compare:nNnT { \l__enumext_level_int } = { 0 }
2440        {
2441          \cs_if_free:NF \__scontents_anskeyenv_env_begin:
2442            {
2443              \msg_fatal:nnn { enumext } { anskeyenv-defined } { anskeyenv }
2444            }
2445        }
2446    }
```

The function __enumext_undefine_anskey_env: will undefine the environment anskeyenv and will be passed to the function __enumext_execute_after_env: (§11.24) which is executed after the environment in which the key save-ans is active.

```
2447  \cs_new_protected:Nn \__enumext_undefine_anskey_env:
2448    {
2449      \cs_undefine:N \anskeyenv
2450      \cs_undefine:N \endanskeyenv
2451      \cs_undefine:N \__scontents_anskeyenv_env_begin:
2452      \cs_undefine:N \__scontents_anskeyenv_env_end:
2453    }
```

(*End of definition for* __enumext_undefine_anskey_env:.)

__enumext_anskey_env_make:n The function __enumext_anskey_env_make:n creates the environment anskeyenv (*custom version* of
__enumext_anskey_env_make:V scontents environment) by setting the initial keys store-env={⟨*store name*⟩} and print-env=false.
anskeyenv To maintain the *scope* of the environment and that it is only active when the key save-ans is active we
__enumext_anskey_env_define_keys: will pass this function to the function __enumext_storing_exec: (§11.23.1).
__enumext_rescan_anskey_env:n

```
2454  \cs_new_protected:Npn \__enumext_anskey_env_make:n #1
2455    {
```

```
2456        \newenvsc{anskeyenv}[store-env=#1,print-env=false]
2457        \__enumext_anskey_env_exec:
2458     }
2459  \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.

```
2460  \cs_new_protected:Nn \__enumext_anskey_env_define_keys:
2461     {
2462        \keys_define:nn { scontents / scontents }
2463          {
2464            break-col .bool_gset:N = \g__enumext_store_columns_break_bool,
2465            break-col .default:n  = true,
2466            break-col .value_forbidden:n = true,
2467            item-join .int_gset:N = \g__enumext_store_item_join_int,
2468            item-join .value_required:n = true,
2469            item-star .bool_gset:N = \g__enumext_store_item_star_bool,
2470            item-star .default:n  = true,
2471            item-star .value_forbidden:n = true,
2472            item-sym* .tl_gset:N   = \g__enumext_store_item_symbol_tl,
2473            item-sym* .value_required:n = true,
2474            item-pos* .dim_gset:N  = \g__enumext_store_item_symbol_sep_dim,
2475            item-pos* .value_required:n = true,
2476            print-env .undefine:,
2477            store-env .undefine:,
2478            write-out .undefine:,
2479          }
2480     }
```

The function __enumext_anskey_env_undefine_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.

```
2481  \cs_new_protected:Nn \__enumext_anskey_env_undefine_keys:
2482     {
2483        \keys_define:nn { scontents / scontents }
2484          {
2485            break-col .undefine:,
2486            item-join .undefine:,
2487            item-star .undefine:,
2488            item-sym* .undefine:,
2489            item-pos* .undefine:,
2490          }
2491     }
```

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*.

```
2492  \cs_new_protected:Npn \__enumext_rescan_anskey_env:n #1
2493     {
2494        \group_begin:
2495          \int_set:Nn \tex_newlinechar:D { `\^^J }
2496          \__scontents_rescan_tokens:x
2497            {
2498              \endgroup % This assumes \catcode`\\=0... Things might go off otherwise.
2499              #1
2500            }
2501     }
```

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

__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*⟩.

```
2502  \cs_new_protected:Nn \__enumext_anskey_env_exec:
2503     {
2504        \__enumext_before_env:nn { anskeyenv }
2505          {
2506            \__enumext_anskey_env_define_keys:
2507          }
```

Now we will execute our actions after the anskeyenv 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.

```
2508      \__enumext_after_env:nn { anskeyenv }
2509        {
2510          \tl_clear:N  \l__enumext_store_anskey_env_tl
2511          \tl_clear:N  \l__enumext_store_anskey_opt_tl
2512          \seq_gpop_right:cNT
2513            { g__scontents_name_ \g__enumext_store_name_tl _seq } \l__enumext_store_anskey_env_tl
2514            { \seq_item:ce { g__scontents_name_ \g__enumext_store_name_tl _seq } { -1 }  }
2515          \__enumext_anskey_env_keys:
2516          \__enumext_anskey_env_store:
2517          \__enumext_anskey_env_clean:
2518          \__enumext_anskey_env_undefine_keys:
2519        }
2520      }
```

💣 Using \seq_gpop_right:cNT is correct here, while we have control of the scope of the environment, we cannot know if a user is using the key store-env={⟨*store name*⟩} in some other environment. The last function __enumext_anskey_env_undefine_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_keys:
__enumext_anskey_env_store:
__enumext_anskey_env_clean:

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.

```
2521  \cs_new_protected:Nn \__enumext_anskey_env_keys:
2522    {
2523      \bool_lazy_and:nnT
2524        { \bool_if_p:N \g__enumext_store_columns_break_bool }
2525        { \bool_not_p:n { \l__enumext_starred_bool } }
2526        {
2527          \tl_put_left:Ne \l__enumext_store_anskey_opt_tl { ,break-col, }
2528        }
```

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

```
2529      \bool_lazy_and:nnT
2530        { \bool_not_p:n { \l__enumext_starred_bool } }
2531        { \int_compare_p:nNn { \g__enumext_store_item_join_int } > { 1 } }
2532        {
2533          \tl_put_left::Ne \l__enumext_store_anskey_opt_tl
2534            {
2535              ,item-join = \exp_not:V \g__enumext_store_item_join_int,
2536            }
2537        }
```

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

```
2538      \bool_if:NT \g__enumext_store_item_star_bool
2539        {
2540          \tl_put_left:Ne \l__enumext_store_anskey_opt_tl
2541            {
2542              ,item-star,
2543            }
2544          \tl_if_empty:NF \g__enumext_store_item_symbol_tl
2545            {
2546              \tl_put_left:Ne \l__enumext_store_anskey_opt_tl
2547                {
2548                  ,item-sym* = \exp_not:V \g__enumext_store_item_symbol_tl,
2549                }
2550            }
2551          \dim_compare:nT
2552            {
2553              \g__enumext_store_item_symbol_sep_dim != \c_zero_dim
2554            }
2555            {
2556              \tl_put_left:Ne \l__enumext_store_anskey_opt_tl
2557                {
2558                  ,item-pos* = \exp_not:V \g__enumext_store_item_symbol_sep_dim,
2559                }
```

```
2560                   }
2561             }
2562       }
```

The function `\__enumext_anskey_env_store:` will be responsible for storing the content of the environment, we will execute the code within a group and only if the variable `\l__enumext_store_-anskey_env_tl` is not empty using the function `\__enumext_rescan_anskey_env:n` from package scontents.

```
2563  \cs_new_protected:Nn \__enumext_anskey_env_store:
2564    {
2565      \group_begin:
2566        \tl_if_empty:NF \l__enumext_store_anskey_env_tl
2567          {
2568            \tl_if_empty:NTF \l__enumext_store_anskey_opt_tl
2569              {
2570                \exp_args:Ne
2571                  \anskey
2572                    {
2573                      \__enumext_rescan_anskey_env:n { \l__enumext_store_anskey_env_tl }
2574                    }
2575              }
2576              {
2577                \keys_set:nV { enumext / anskey } \l__enumext_store_anskey_opt_tl
2578                \exp_args:Ne
2579                  \anskey
2580                    {
2581                      \__enumext_rescan_anskey_env:n { \l__enumext_store_anskey_env_tl }
2582                    }
2583              }
2584          }
2585      \group_end:
2586    }
```

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

```
2587  \cs_new_protected:Nn \__enumext_anskey_env_clean:
2588    {
2589      \bool_gset_false:N \g__enumext_store_columns_break_bool
2590      \int_gzero:N       \g__enumext_store_item_join_int
2591      \bool_gset_false:N \g__enumext_store_item_star_bool
2592      \tl_gclear:N       \g__enumext_store_item_symbol_tl
2593      \dim_gzero:N       \g__enumext_store_item_symbol_sep_dim
2594    }
```

(*End of definition for* `\__enumext_anskey_env_keys:` *,* `\__enumext_anskey_env_store:` *, and* `\__enumext_anskey_env_-clean:` *.*)

11.28   Common functions for keyans, keyans* and keyanspic

11.28.1   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`.

```
2595  \cs_new_protected:Npn \__enumext_keyans_addto_prop:n #1
2596    {
2597      \tl_clear:N \l__enumext_store_current_label_tl
2598      \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
2599        {
2600          \tl_put_right:Ne \l__enumext_store_current_label_tl { \l__enumext_label_vi_tl }
2601        }
2602        {
2603          \tl_put_right:Ne \l__enumext_store_current_label_tl { \l__enumext_label_v_tl }
2604        }
2605      \tl_if_novalue:nF { #1 }
2606        {
2607          % Set save-sep
2608          \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_tl
2609            {
```

```
2610              \tl_put_right:Ne \l__enumext_store_current_label_tl { \l__enumext_store_keyans_item_o
2611            }
2612          \tl_put_right:Ne \l__enumext_store_current_label_tl { #1 }
2613        }
2614      \__enumext_store_addto_prop:V \l__enumext_store_current_label_tl
2615    }
```

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

11.28.2 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.

```
2616 \cs_new_protected:Nn \__enumext_keyans_store_ref:
2617    {
2618      \bool_if:NT \l__enumext_store_ref_key_bool
2619        {
2620          \cs_set_protected:Npn \__enumext_tmp:n ##1
2621            {
2622              \tl_set_eq:cc { l__enumext_label_copy_##1_tl } { l__enumext_label_##1_tl }
2623              \tl_reverse:c { l__enumext_label_copy_##1_tl }
2624              \tl_remove_once:cn { l__enumext_label_copy_##1_tl } { . }
2625              \tl_reverse:c { l__enumext_label_copy_##1_tl }
2626            }
2627          \clist_map_inline:nn { i, v, vi, vii, viii } { \__enumext_tmp:n {##1} }
2628          \__enumext_keyans_store_ref_aux_i:
2629        }
2630    }
```

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.

```
2631 \cs_new_protected:Nn \__enumext_keyans_store_ref_aux_i:
2632    {
2633      \bool_if:NT \g__enumext_starred_bool
2634        {
2635          \tl_set_eq:NN \l__enumext_label_copy_i_tl \l__enumext_label_copy_vii_tl
2636        }
2637      \int_compare:nNnT { \l__enumext_keyans_pic_level_int } = { 1 }
2638        {
2639          \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2640            { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_vi_tl }
2641        }
2642      \int_compare:nNnT { \l__enumext_keyans_level_int } = { 1 }
2643        {
2644          \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2645            { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_v_tl }
2646        }
2647      \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
2648        {
2649          \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2650            { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_viii_tl }
2651        }
2652      \tl_put_right:Ne \l__enumext_newlabel_arg_one_tl
2653        {
2654          \l__enumext_store_name_tl \c_colon_str
2655          \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } }
2656        }
2657      \__enumext_keyans_store_ref_aux_ii:
2658    }
```

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.

```
2659 \cs_new_protected:Nn \__enumext_keyans_store_ref_aux_ii:
2660    {
2661      \tl_put_right:Ne \l__enumext_write_aux_file_tl
```

```
2662        {
2663            \__enumext_newlabel:nn
2664                { \exp_not:V \l__enumext_newlabel_arg_one_tl }
2665                { \l__enumext_newlabel_arg_two_tl }
2666        }
2667        \l__enumext_write_aux_file_tl
2668    }
```

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

11.28.3   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.

```
2669 \cs_new_protected:Npn \__enumext_keyans_addto_seq:n #1
2670    {
2671        \tl_clear:N \l__enumext_store_current_label_tl
2672        \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
2673            {
2674                \tl_put_right:Ne \l__enumext_store_current_label_tl { \item \l__enumext_label_vi_tl }
2675            }
2676            {
2677                \tl_put_right:Ne \l__enumext_store_current_label_tl { \item \l__enumext_label_v_tl }
2678            }
2679        \tl_if_novalue:nF { #1 }
2680            {
2681                \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_tl
2682                    {
2683                        \tl_put_right:Ne \l__enumext_store_current_label_tl
2684                            {
2685                                \l__enumext_store_keyans_item_opt_sep_tl
2686                            }
2687                    }
2688                \tl_put_right:Ne \l__enumext_store_current_label_tl { #1 }
2689            }
2690        \__enumext_keyans_addto_seq_link:
2691    }
```

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.

```
2692 \cs_new_protected:Nn \__enumext_keyans_addto_seq_link:
2693    {
2694        \bool_lazy_and:nnT
2695            { \bool_if_p:N \l__enumext_store_ref_key_bool }
2696            { \bool_if_p:N \l__enumext_hyperref_bool }
2697            {
2698                \tl_put_right:Ne \l__enumext_store_current_label_tl
2699                    {
2700                        \hfill \exp_not:N \hyperlink
2701                            {
2702                                \exp_not:V \l__enumext_newlabel_arg_one_tl
2703                            }
2704                            { \exp_not:V \l__enumext_mark_ref_sym_tl }
2705                    }
2706            }
2707        \__enumext_store_addto_seq:V \l__enumext_store_current_label_tl
2708        \bool_if:NT \l__enumext_check_answers_bool
2709            {
2710                \int_gincr:N \g__enumext_item_anskey_int
2711            }
2712    }
```

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

11.28.4 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.

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.

```
\__enumext_keyans_show_left:n
\__enumext_keyans_show_ans:
\__enumext_keyans_show_pos:
\__enumext_keyans_show_item_opt:
```

```
2713 \cs_new_protected:Npn \__enumext_keyans_show_left:n #1
2714   {
2715     \tl_if_novalue:nF { #1 }
2716       {
2717         \tl_set:Ne \l__enumext_store_current_opt_arg_tl { #1 }
2718       }
2719     \bool_if:NT \l__enumext_show_answer_bool
2720       {
2721         \__enumext_keyans_show_ans:
2722       }
2723     \bool_if:NT \l__enumext_show_position_bool
2724       {
2725         \__enumext_keyans_show_pos:
2726       }
2727   }
2728 \cs_new_protected:Nn \__enumext_keyans_show_item_opt:
2729   {
2730     \tl_if_empty:NF \l__enumext_store_current_opt_arg_tl
2731       {
2732         \bool_lazy_or:nnT
2733           { \bool_if_p:N \l__enumext_show_answer_bool }
2734           { \bool_if_p:N \l__enumext_show_position_bool }
2735           {
2736             \__enumext_keyans_wrapper_opt:n { \l__enumext_store_current_opt_arg_tl } \c_space_tl
2737           }
2738       }
2739   }
2740 \cs_new_protected:Nn \__enumext_keyans_show_ans:
2741   {
2742     \tl_put_left:Nn \l__enumext_label_v_tl
2743       {
2744         \__enumext_print_keyans_box:NN
2745           \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
2746       }
2747   }
2748 \cs_new_protected:Nn \__enumext_keyans_show_pos:
2749   {
2750     \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
2751       {
2752         \tl_set:Ne \l__enumext_mark_answer_sym_tl
2753           {
2754             \group_begin:
2755             \exp_not:N \normalfont
2756             \exp_not:N \footnotesize [ \int_eval:n
2757               {
2758                 \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
2759               }
2760             ]
2761             \group_end:
2762           }
2763       }
2764       {
2765         \tl_set:Ne \l__enumext_mark_answer_sym_tl
2766           {
2767             \group_begin:
2768             \exp_not:N \normalfont
2769             \exp_not:N \footnotesize [ \int_eval:n
2770               {
2771                 \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } + 1
2772               }
2773             ]
2774             \group_end:
2775           }
2776       }
2777     \tl_put_left:Nn \l__enumext_label_v_tl
```

```
2778        {
2779            \__enumext_print_keyans_box:NN
2780                \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
2781        }
2782    }
```

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

11.29 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*`

```
2783  \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
2784    {
2785      \keys_define:nn { enumext / #1 }
2786        {
2787          item-sym* .tl_set:c  = { l__enumext_item_symbol_#2_tl },
2788          item-sym* .value_required:n = true,
2789          item-sym* .initial:n  = {$\star$},
2790          item-pos* .dim_set:c  = { l__enumext_item_symbol_sep_#2dim },
2791          item-pos* .value_required:n = true,
2792        }
2793    }
2794  \clist_map_inline:nn
2795    {
2796      {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {enumext*}{vii}
2797    }
2798    { \__enumext_tmp:nn #1 }
```

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

11.30 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.

```
2799  \cs_new_protected:Nn \__enumext_footnotetext:nn
2800    {
2801      \footnotetext[#1]{#2}
2802    }
2803  \cs_new_protected:Nn \__enumext_renew_footnote:
2804    {
2805      \seq_gclear:N \g__enumext_footnote_arg_seq
2806      \seq_gclear:N \g__enumext_footnote_int_seq
2807      \RenewDocumentCommand \footnote { o +m }
2808        {
2809          \tl_if_novalue:nTF {##1}
2810            {
2811              \stepcounter{footnote}
2812              \int_gset_eq:Nc \g__enumext_footnote_int { c@footnote }
2813            }
2814            {
2815              \int_gset:Nn \g__enumext_footnote_int { ##1 }
2816            }
2817          \footnotemark [ \g__enumext_footnote_int ]
2818          \seq_gput_right:Nn \g__enumext_footnote_arg_seq { ##2 }
2819          \seq_gput_right:NV \g__enumext_footnote_int_seq \g__enumext_footnote_int
2820        }
2821    }
2822  \cs_new_protected:Nn \__enumext_print_footnote:
2823    {
2824      \seq_if_empty:NF \g__enumext_footnote_int_seq
2825        {
2826          \seq_map_pairwise_function:NNN
2827            \g__enumext_footnote_int_seq
2828            \g__enumext_footnote_arg_seq
2829            \__enumext_footnotetext:nn
2830        }
2831    }
```

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

11.31   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.

11.31.1   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: (§11.32).

```
2832  \cs_new_protected:Npn \__enumext_default_item:n #1
2833    {
2834      \tl_if_novalue:nTF {#1}
2835        {
2836          \bool_if:NT \l__enumext_check_answers_bool
2837            {
2838              \int_gincr:N \g__enumext_item_number_int
2839            }
2840          \bool_set_true:c { l__enumext_wrap_label_ \__enumext_level: _bool }
2841          \__enumext_item_std:w \tl_use:c { l__enumext_fake_item_indent_ \__enumext_level: _tl }
2842        }
2843        {
2844          \bool_set_eq:cc
2845            { l__enumext_wrap_label_ \__enumext_level: _bool }
2846            { l__enumext_wrap_label_opt_ \__enumext_level: _bool }
2847          \__enumext_item_std:w [#1] \tl_use:c { l__enumext_fake_item_indent_ \__enumext_level: _tl
2848        }
2849    }
```

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

__enumext_starred_item:nn

The \item*, \item*[⟨symbol⟩] and \item*[⟨symbol⟩][⟨offset⟩] works like the numbered \item, but placing a [⟨symbol⟩] to the *"left"* of the ⟨label⟩ separated from it by the value set by the labelsep key and can be *offset* using the second optional argument [⟨offset⟩].
#1: \l__enumext_item_symbol_X_tl
#2: \l__enumext_item_symbol_sep_X_dim
First we will make a copy of \l__enumext_item_symbol_X_tl which is set by the key item-sym* or passed as optional argument in the global variable \g__enumext_item_symbol_tl, followed by setting the variable \l__enumext_item_symbol_sep_X_dim set by the key item*-sep or by the second optional argument.

Then we will see the state of the variable \l__enumext_check_ans_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: (§11.32).

```
2850  \cs_new_protected:Npn \__enumext_starred_item:nn #1 #2
2851    {
2852      \tl_if_novalue:nF {#1}
2853        {
2854          \tl_set:cn { l__enumext_item_symbol_ \__enumext_level: _tl } {#1}
2855        }
2856      \tl_gset_eq:Nc \g__enumext_item_symbol_tl { l__enumext_item_symbol_ \__enumext_level: _tl }
2857      \tl_if_novalue:nTF {#2}
2858        {
2859          \dim_set_eq:cc
2860            { l__enumext_item_symbol_sep_ \__enumext_level: _dim }
2861            { l__enumext_labelsep_ \__enumext_level: _dim }
2862        }
2863        {
```

```
2864            \dim_set:cn { l__enumext_item_symbol_sep_ \__enumext_level: _dim } {#2}
2865        }
2866      \bool_if:NT \l__enumext_check_answers_bool
2867        {
2868          \int_gincr:N \g__enumext_item_number_int
2869        }
2870      \bool_set_true:c { l__enumext_wrap_label_ \__enumext_level: _bool }
2871      \__enumext_item_std:w \tl_use:c { l__enumext_fake_item_indent_ \__enumext_level: _tl }
2872    }
```

(*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 (§11.33.2).

```
2873 \cs_new_protected:Nn \__enumext_redefine_item:
2874    {
2875      \RenewDocumentCommand \item { s o o }
2876        {
2877         \bool_if:nTF {##1}
2878           {
2879             \__enumext_starred_item:nn {##2} {##3}
2880           }
2881           { \__enumext_default_item:n {##2} }
2882        }
2883    }
```

(*End of definition for* __enumext_redefine_item:.)

11.31.2   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.

```
2884 \cs_new_protected:Npn \__enumext_keyans_default_item:n #1
2885    {
2886      \tl_if_novalue:nTF { #1 }
2887        {
2888          \bool_set_true:N \l__enumext_wrap_label_v_bool
2889          \__enumext_item_std:w \tl_use:N \l__enumext_fake_item_indent_v_tl
2890        }
2891        {
2892          \bool_set_eq:NN \l__enumext_wrap_label_v_bool \l__enumext_wrap_label_opt_v_bool
2893          \__enumext_item_std:w [#1] \tl_use:N \l__enumext_fake_item_indent_v_tl
2894        }
2895    }
```

(*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⟩.

```
2896 \cs_new_protected:Npn \__enumext_keyans_starred_item:n #1
2897    {
2898      \tl_set_eq:NN \l__enumext_store_current_label_tmp_tl \l__enumext_label_v_tl
2899      \__enumext_keyans_show_left:n { #1 }
2900      \bool_set_true:N \l__enumext_wrap_label_v_bool
2901      \__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⟩.

```
2902      \tl_set_eq:NN \l__enumext_label_v_tl \l__enumext_store_current_label_tmp_tl
2903      \__enumext_keyans_addto_prop:n { #1 }
2904      \__enumext_keyans_store_ref:
2905      \__enumext_keyans_addto_seq:n { #1 }
2906      \int_gincr:N \g__enumext_check_starred_cmd_int
2907    }
```

(*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 (§11.33.2).

```
2908  \cs_new_protected:Nn \__enumext_keyans_redefine_item:
2909    {
2910      \RenewDocumentCommand \item { s o }
2911        {
2912          \bool_if:nTF {##1}
2913            {
2914              \peek_remove_spaces:n
2915                {
2916                  \__enumext_keyans_starred_item:n {##2}
2917                }
2918            }
2919            {
2920              \__enumext_keyans_default_item:n {##2}
2921            }
2922        }
2923    }
```

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

11.32   Redefining \makelabel **command**

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

11.32.1   Redefining \makelabel **for enumext**

__enumext_item_starred:

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

```
2924  \cs_new_protected:Nn \__enumext_item_starred:
2925    {
2926      \tl_if_empty:cF { l__enumext_item_symbol_ \__enumext_level: _tl }
2927        {
2928          \mode_leave_vertical:
2929          \skip_horizontal:n { -\dim_use:c { l__enumext_item_symbol_sep_ \__enumext_level: _dim } }
2930          \makebox[ 0pt ][ r ]{ \g__enumext_item_symbol_tl }
2931          \skip_horizontal:n { \dim_use:c { l__enumext_item_symbol_sep_ \__enumext_level: _dim } }
2932        }
2933    }
```

(*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 (§11.33.2).

```
2934  \cs_new_protected:Nn \__enumext_make_label:
2935    {
2936      \RenewDocumentCommand \makelabel { m }
2937        {
2938          \tl_use:c { l__enumext_label_fill_left_ \__enumext_level: _tl }
2939          \tl_use:c { l__enumext_label_font_style_ \__enumext_level: _tl }
2940          \bool_if:cTF { l__enumext_wrap_label_ \__enumext_level: _bool }
2941            {
2942              \__enumext_item_starred:
2943              \use:c { __enumext_wrapper_label_ \__enumext_level: :n } { ##1 }
2944            }
2945            { ##1 }
2946          \tl_use:c { l__enumext_label_fill_right_ \__enumext_level: _tl }
2947          \tl_gclear:N \g__enumext_item_symbol_tl
2948        }
2949    }
```

(*End of definition for* __enumext_make_label:.)

11.32.2 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 (§11.33.2).

```
2950  \cs_new_protected:Nn \__enumext_keyans_make_label:
2951    {
2952      \RenewDocumentCommand \makelabel { m }
2953        {
2954          \tl_use:N \l__enumext_label_fill_left_v_tl
2955          \tl_use:N \l__enumext_label_font_style_v_tl
2956          \bool_if:NTF \l__enumext_wrap_label_v_bool
2957            {
2958              \__enumext_wrapper_label_v:n { ##1 }
2959            }
2960            { ##1 }
2961          \tl_use:N \l__enumext_label_fill_right_v_tl
2962        }
2963    }
```

(*End of definition for* __enumext_keyans_make_label:.)

11.33 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.

11.33.1 Calculation of \leftmargin and \itemindent

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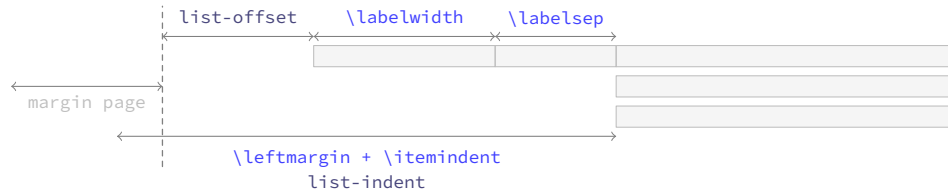
Figure 9: Representation of standard horizontal lengths in list environment.

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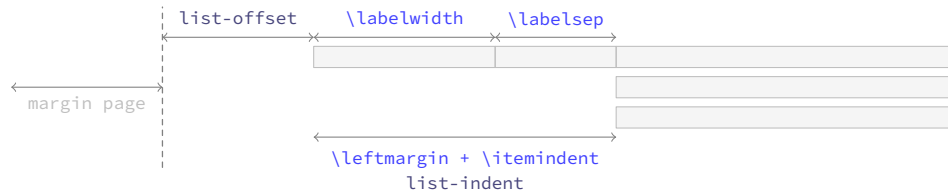
Figure 10: Representation of horizontal lengths concept in list in enumext.

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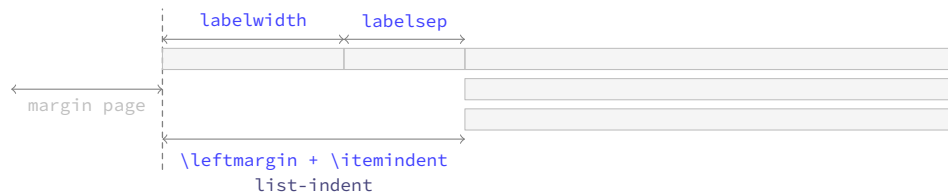
Figure 11: Default horizontal lengths in enumext.

__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 (§11.33.2).

```
2964  \cs_new_protected:Npn \__enumext_calc_hspace:NNNNNNN #1 #2 #3 #4 #5 #6 #7
2965    {
2966      \dim_compare:nNnT { #1 } < { \c_zero_dim }
2967        {
2968          \msg_warning:nnnV { enumext } { width-non-positive }{ labelwidth }{ #1 }
2969          \dim_set:Nn #1 { \dim_abs:n { #1 } }
2970        }
2971      \dim_compare:nNnT { #2 } < { \c_zero_dim }
2972        {
2973          \msg_warning:nnnV { enumext } { width-negative }{ labelsep }{ #2 }
2974          \dim_set:Nn #2 { \dim_abs:n { #2 } }
2975        }
```

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

```
2976      \bool_if:nF #7 { \dim_set:Nn #4 { #1 + #2} }
```

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

```
2977      \dim_compare:nNnTF { #4 } < { \c_zero_dim }
2978        {
2979          \dim_set:Nn #6 { #1 + #2 - #4}
2980          \dim_set:Nn #5 { #1 + #2 + #3 - #6 }
2981        }
2982        {
2983          \dim_compare:nNnT { #4 } = { #1 + #2 }
2984            { \dim_set:Nn #6 { \c_zero_dim } }
2985          \dim_compare:nNnT { #4 } < { #1 + #2 }
2986            { \dim_set:Nn #6 { #1 + #2 - #4} }
2987          \dim_compare:nNnT { #4 } > { #1 + #2 }
2988            {
2989              \dim_set:Nn #6 { -#1 - #2 + #4}
2990              \dim_set:Nn #6 { #6*-1}
2991            }
2992          \dim_set:Nn #5 { #1 + #2 + #3 - #6 }
2993        }
2994    }
2995  \cs_generate_variant:Nn \__enumext_calc_hspace:NNNNNNN { ccccccc }
```

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

11.33.2 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.

```
2996  \cs_set_protected:Npn \__enumext_tmp:n #1
2997    {
2998      \cs_new_protected:cpn { __enumext_list_arg_two_#1: }
2999        {
3000          \__enumext_calc_hspace:ccccccc
3001            { l__enumext_labelwidth_#1_dim } { l__enumext_labelsep_#1_dim }
3002            { l__enumext_listoffset_#1_dim } { l__enumext_leftmargin_tmp_#1_dim }
3003            { l__enumext_leftmargin_#1_dim } { l__enumext_itemindent_#1_dim }
3004            { l__enumext_leftmargin_tmp_#1_bool }
3005          \clist_map_inline:nn
3006            { labelsep, labelwidth, itemindent, leftmargin, rightmargin, listparindent }
3007            { \dim_set_eq:cc {####1} { l__enumext_####1_#1_dim } }
3008          \clist_map_inline:nn { topsep, parsep, partopsep, itemsep }
3009            { \skip_set_eq:cc {####1} { l__enumext_####1_#1_skip } }
3010          \usecounter { enumX#1 }
3011          \setcounter { enumX#1 } { \int_eval:n { \int_use:c { l__enumext_start_#1_int } - 1 } }
3012          \str_if_eq:nnTF {#1} { v }
3013            {
3014              \__enumext_keyans_redefine_item:
3015              \__enumext_keyans_make_label:
3016              \__enumext_keyans_ref:
3017              \__enumext_keyans_fake_item:
3018              \bool_if:cT { l__enumext_show_length_#1_bool }
3019                {
```

```
3020              \msg_term:nnnn { enumext } { list-lengths-not-nested } { v } { keyans }
3021            }
3022          }
3023          {
3024            \__enumext_redefine_item:
3025            \__enumext_make_label:
3026            \__enumext_standar_ref:
3027            \__enumext_fake_item:
3028            \bool_if:cT { l__enumext_show_length_#1_bool }
3029              {
3030                \msg_term:nnne { enumext } { list-lengths } {#1} { \int_use:N \l__enumext_level_i
3031              }
3032          }
3033        }
3034      }
3035    \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.

```
3036    \cs_set_protected:Npn \__enumext_tmp:n #1
3037      {
3038        \cs_new_protected:cpn { __enumext_list_arg_two_#1: }
3039          {
3040            \__enumext_calc_hspace:ccccccc
3041              { l__enumext_labelwidth_#1_dim } { l__enumext_labelsep_#1_dim }
3042              { l__enumext_listoffset_#1_dim } { l__enumext_leftmargin_tmp_#1_dim }
3043              { l__enumext_leftmargin_#1_dim } { l__enumext_itemindent_#1_dim }
3044              { l__enumext_leftmargin_tmp_#1_bool }
3045            \clist_map_inline:nn
3046              { labelsep, labelwidth, itemindent, leftmargin, rightmargin, listparindent }
3047              { \dim_set_eq:cc {####1} { l__enumext_####1_#1_dim } }
3048            \clist_map_inline:nn { topsep, parsep, partopsep, itemsep }
3049              { \skip_set_eq:cc {####1} { l__enumext_####1_#1_skip } }
3050            \skip_set_eq:Nc \parsep  { l__enumext_itemsep_#1_skip }
3051            \skip_zero:N \partopsep
3052            \usecounter { enumX#1 }
3053            \setcounter { enumX#1 } { \int_eval:n { \int_use:c { l__enumext_start_#1_int } - 1 } }
3054            \__enumext_starred_ref:
3055            \str_if_eq:nnTF {#1} { vii }
3056              {
3057                \__enumext_fake_item_vii:
3058                \bool_if:cT { l__enumext_show_length_vii_bool }
3059                  { \msg_term:nnnn { enumext } { list-lengths-not-nested } { vii } { enumext* } }
3060              }
3061              {
3062                \__enumext_fake_item_viii:
3063                \bool_if:cT { l__enumext_show_length_#1_bool }
3064                  { \msg_term:nnnn { enumext } { list-lengths-not-nested } { #1 } { keyans* } }
3065              }
3066          }
3067      }
3068    \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:`.)

11.34   The environment enumext

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

```
3069    \NewDocumentEnvironment{enumext}{ O{} }
3070      {
3071        \__enumext_safe_exec:
3072        \__enumext_parse_keys:n {#1}
3073        \__enumext_before_list:
3074        \__enumext_start_store_level:
3075        \__enumext_start_list:nn
3076          { \tl_use:c { l__enumext_label_ \__enumext_level: _tl } }
3077          {
3078            \use:c { __enumext_list_arg_two_ \__enumext_level: : }
```

```
3079        \__enumext_before_keys_exec:
3080      }
3081    \__enumext_after_args_exec:
3082  }
3083  {
3084    \__enumext_stop_list:
3085    \__enumext_stop_store_level:
3086    \__enumext_after_list:
3087  }
```

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

__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:.

```
3088  \cs_new_protected:Nn \__enumext_safe_exec:
3089    {
3090      \__enumext_internal_mini_page:
3091      \__enumext_is_not_nested:
3092      \int_incr:N \l__enumext_level_int
3093      \int_compare:nNnT { \l__enumext_level_int } > { 4 }
3094        { \msg_fatal:nn { enumext } { list-too-deep } }
3095      \bool_set_true:N \l__enumext_standar_bool
3096      \__enumext_is_on_first_level:
3097    }
```

(*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.

```
3098  \cs_new_protected:Npn \__enumext_parse_keys:n #1
3099    {
3100      \tl_if_novalue:nF {#1}
3101        {
3102          \str_clear:N \l__enumext_series_str
3103          \int_compare:nNnTF { \l__enumext_level_int } = { 1 }
3104            {
3105              \keys_set:nn { enumext / level-1 } {#1}
3106              \__enumext_parse_series:n {#1}
3107            }
3108            {
3109              \exp_args:Ne \keys_set:nn
3110                { enumext / level-\int_use:N \l__enumext_level_int } {#1}
3111            }
3112          \__enumext_store_active_keys:n {#1}
3113        }
3114    }
```

(*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*⟩ of the \anskey command.
If enumext are nested in enumext* add __enumext_store_level_open: to preserve the stored structure.

```
3115  \cs_new_protected:Nn \__enumext_start_store_level:
3116    {
3117      \bool_lazy_all:nT
3118        {
3119          { \bool_if_p:N \l__enumext_store_active_bool }
3120          { \bool_not_p:n { \l__enumext_keyans_env_bool } }
3121          { \bool_not_p:n { \g__enumext_starred_bool } }
3122        }
3123        {
```

```
3124                \int_compare:nNnT { \l__enumext_level_int } > { 1 }
3125                  {
3126                    \bool_set_true:c { l__enumext_store_upper_level_ \__enumext_level: _bool }
3127                    \__enumext_store_level_open:
3128                  }
3129              }
3130          \bool_lazy_all:nT
3131            {
3132                { \bool_if_p:N \l__enumext_store_active_bool }
3133                { \bool_not_p:n { \l__enumext_keyans_env_bool } }
3134                { \bool_if_p:N \g__enumext_starred_bool }
3135            }
3136            {
3137                \int_compare:nNnT { \l__enumext_level_int } > { 0 }
3138                  {
3139                    \bool_set_true:c { l__enumext_store_upper_level_ \__enumext_level: _bool }
3140                    \__enumext_store_level_open:
3141                  }
3142            }
3143        }
3144    \cs_new_protected:Nn \__enumext_stop_store_level:
3145      {
3146        \bool_if:cT { l__enumext_store_upper_level_ \__enumext_level: _bool }
3147          {
3148            \__enumext_store_level_close:
3149          }
3150      }
```

(*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.

```
3151    \cs_new_protected:Nn \__enumext_before_list:
3152      {
3153        \__enumext_vspace_above:
3154        \__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.

```
3155        \__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.

```
3156        \dim_compare:nNnT
3157          { \dim_use:c { l__enumext_minipage_right_ \__enumext_level: _dim } } > { \c_zero_dim }
3158          {
3159            \dim_set:cn { l__enumext_minipage_left_ \__enumext_level: _dim }
3160              {
3161                \linewidth
3162                - \dim_use:c { l__enumext_minipage_right_ \__enumext_level: _dim }
3163                - \dim_use:c { l__enumext_minipage_hsep_ \__enumext_level: _dim }
3164              }
```

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*.

```
3165            \bool_set_true:c { l__enumext_minipage_active_ \__enumext_level: _bool }
3166            \int_gincr:N \g__enumext_minipage_stat_int
3167            \__enumext_mini_addvspace:
3168            \nointerlineskip\noindent
3169            \begin{__enumext_mini_env*}
3170              { \dim_use:c { l__enumext_minipage_left_ \__enumext_level: _dim } }
3171          }
3172        \__enumext_multicols_start:
```

```
3173       }
```

(*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.

```
3174 \cs_new_protected:Nn \__enumext_multicols_start:
3175   {
3176     \int_compare:nNnT
3177       { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } > { 1 }
3178       {
3179         \dim_compare:nNnT
3180           { \dim_use:c { l__enumext_columns_sep_ \__enumext_level: _dim } } = { \c_zero_dim }
3181           {
3182             \dim_set:cn { l__enumext_columns_sep_ \__enumext_level: _dim }
3183               {
3184                 ( \dim_use:c { l__enumext_labelwidth_ \__enumext_level: _dim }
3185                   + \dim_use:c { l__enumext_labelsep_ \__enumext_level: _dim }
3186                 ) / \int_use:c { l__enumext_columns_ \__enumext_level: _int }
3187                 - \dim_use:c { l__enumext_listoffset_ \__enumext_level: _dim }
3188               }
3189           }
3190         \dim_set_eq:Nc \columnsep { l__enumext_columns_sep_ \__enumext_level: _dim  }
3191         \skip_zero:N \multicolsep
3192         \int_compare:nNnT { \l__enumext_level_int } > { 1 }
3193           {
3194             \dim_zero:N \columnseprule
3195           }
```

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.

```
3196         \bool_if:cF { l__enumext_minipage_active_ \__enumext_level: _bool }
3197           {
3198             \__enumext_multi_addvspace:
3199           }
3200         \raggedcolumns
3201         \begin{multicols}{ \int_use:c { l__enumext_columns_ \__enumext_level: _int } }
3202       }
3203   }
```

(*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.

```
3204 \cs_new_protected:Nn \__enumext_multicols_stop:
3205   {
3206     \int_compare:nNnT
3207       { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } > { 1 }
3208       {
3209         \end{multicols}
3210         \bool_if:cF { l__enumext_minipage_active_ \__enumext_level: _bool }
3211           {
3212             \par\addvspace{ \skip_use:c { l__enumext_multicols_below_ \__enumext_level: _skip } }
3213           }
3214       }
3215   }
```

(*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.

```
3216 \cs_new_protected:Nn \__enumext_after_list:
3217   {
```

```
3218      \bool_if:cTF { l__enumext_minipage_active_ \__enumext_level: _bool }
3219        {
3220          \int_compare:nNnT { \g__enumext_minipage_stat_int } = { 1 }
3221            {
3222              \msg_warning:nn { enumext } { missing-miniright }
3223              \miniright
3224            }
3225          \int_gzero:N \g__enumext_minipage_stat_int
3226          \end{__enumext_mini_env*}
3227          \par\addvspace { \l__enumext_minipage_after_skip }
3228        }
3229        { \__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.

```
3230      \__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.

```
3231      \__enumext_after_stop_list:
3232      \__enumext_vspace_below:
3233      \bool_set_false:N \l__enumext_standar_bool
3234      \__enumext_resume_save_counter:
3235    }
```

(*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.

```
3236  \__enumext_after_env:nn {enumext} { \__enumext_execute_after_env: }
```

11.35 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.

```
3237  \NewDocumentEnvironment{keyans}{ O{} }
3238    {
3239      \__enumext_keyans_safe_exec:
3240      \__enumext_keyans_parse_keys:n {#1}
3241      \__enumext_before_list_v:
3242      \__enumext_start_list:nn
3243        { \tl_use:N \l__enumext_label_v_tl }
3244        {
3245          \__enumext_list_arg_two_v:
3246          \__enumext_before_keys_exec_v:
3247        }
3248      \__enumext_after_args_exec_v:
3249    }
3250    {
3251      \__enumext_check_starred_cmd:n { item }
3252      \__enumext_stop_list:
3253      \__enumext_after_list_v:
3254    }
```

(*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.

```
3255  \cs_new_protected:Nn \__enumext_keyans_safe_exec:
3256    {
3257      \bool_if:NF \l__enumext_store_active_bool
3258        {
3259          \msg_error:nnnn { enumext } { wrong-place }{ keyans }{ save-ans }
3260        }
3261      \int_incr:N \l__enumext_keyans_level_int
3262      \bool_set_true:N \l__enumext_keyans_env_bool
3263      \__enumext_keyans_start_line:
```

```
3264      % Set false for interfering with enumext nested in keyans (yes, its possible and crayze)
3265      \bool_set_false:N \l__enumext_store_active_bool
3266      \int_compare:nNnT { \l__enumext_keyans_level_int } > { 1 }
3267        {
3268          \msg_error:nn { enumext } { keyans-nested }
3269        }
3270      \int_compare:nNnT { \l__enumext_level_int } > { 1 }
3271        {
3272          \msg_error:nn { enumext } { keyans-wrong-level }
3273        }
3274    }
```

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

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

```
3275 \cs_new_protected:Npn \__enumext_keyans_parse_keys:n #1
3276   {
3277      \keys_set:nn { enumext / keyans } {#1}
3278   }
```

(*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.

```
3279 \cs_new_protected:Nn \__enumext_before_list_v:
3280   {
3281      \__enumext_vspace_above_v:
3282      \__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.

```
3283      \dim_compare:nNnT { \l__enumext_minipage_right_v_dim } > { \c_zero_dim }
3284        {
3285          \dim_set:Nn \l__enumext_minipage_left_v_dim
3286            {
3287              \linewidth - \l__enumext_minipage_right_v_dim - \l__enumext_minipage_hsep_v_dim
3288            }
```

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*.

```
3289          \bool_set_true:N \l__enumext_minipage_active_v_bool
3290          \int_gincr:N \g__enumext_minipage_stat_int
3291          \__enumext_keyans_mini_addvspace:
3292          \nointerlineskip\noindent
3293          \begin{__enumext_mini_env*}{ \l__enumext_minipage_left_v_dim }
3294        }
```

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

```
3295      \__enumext_keyans_multicols_start:
3296   }
```

(*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.

```
3297 \cs_new_protected:Nn \__enumext_keyans_multicols_start:
3298   {
3299      \int_compare:nNnT { \l__enumext_columns_v_int } > { 1 }
3300        {
```

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

```
3301        \dim_compare:nNnT { \l__enumext_columns_sep_v_dim } = { \c_zero_dim }
3302          {
3303            \dim_set:Nn \l__enumext_columns_sep_v_dim
3304              {
3305                (
3306                  \l__enumext_labelwidth_v_dim + \l__enumext_labelsep_v_dim
3307                ) / \l__enumext_columns_v_int
3308                - \l__enumext_listoffset_v_dim
3309              }
3310          }
3311        \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).

```
3312        \skip_zero:N \multicolsep
3313        \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.

```
3314        \bool_if:NF \l__enumext_minipage_active_v_bool
3315          {
3316            \__enumext_keyans_multi_addvspace:
3317          }
3318        \raggedcolumns
3319        \begin{multicols}{ \l__enumext_columns_v_int }
3320      }
3321    }
```

(*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.

```
3322  \cs_new_protected:Nn \__enumext_keyans_multicols_stop:
3323    {
3324      \int_compare:nNnT { \l__enumext_columns_v_int } > { 1 }
3325        {
3326          \end{multicols}
3327          \bool_if:NF \l__enumext_minipage_active_v_bool
3328            {
3329              \par\addvspace{ \l__enumext_multicols_below_v_skip }
3330            }
3331        }
3332    }
```

(*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.

```
3333  \cs_new_protected:Nn \__enumext_after_list_v:
3334    {
3335      \bool_if:NTF \l__enumext_minipage_active_v_bool
3336        {
3337          \int_compare:nNnT { \g__enumext_minipage_stat_int } = { 1 }
3338            {
3339              \msg_warning:nn { enumext } { missing-miniright }
3340              \miniright
3341            }
3342          \int_gzero:N \g__enumext_minipage_stat_int
3343          \end{__enumext_mini_env*}
3344          \par\addvspace{ \l__enumext_minipage_after_skip }
3345        }
3346        { \__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.

```
3347      \bool_set_false:N \l__enumext_keyans_env_bool
3348      \__enumext_after_stop_list_v:
3349      \__enumext_vspace_below_v:
3350    }
```

(*End of definition for* __enumext_after_list_v:.)

11.36 The environment keyanspic and \anspic

The keyanspic environment is a list-based environment that uses the same configuration for *"spacing"* and ⟨*label*⟩ as the keyans environment, but it does not use \item.

The contents are passed to the environment by means of the \anspic command and are placed inside minipage environments, with the ⟨*label*⟩ underneath, adjusting widths according to the options passed to the environment.

Again it is necessary to "adjust" the spacing, both vertical and horizontal, to obtain an output like the one shown in the figure 12.
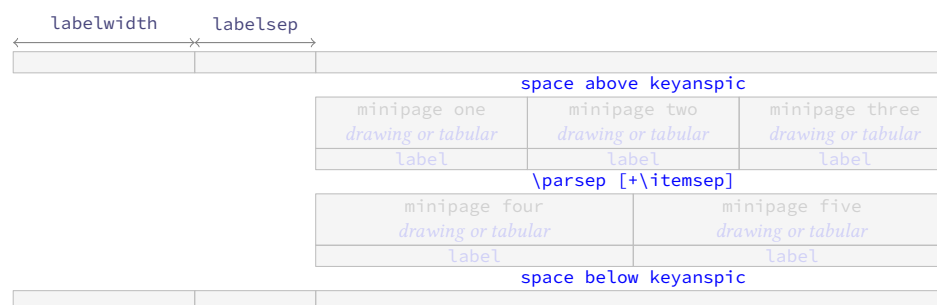
Figure 12: Representation of the keyanspic spacing in enumext.

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

11.36.1 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.

```
3351    \NewDocumentCommand \anspic { s o +m }
3352      {
```

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

```
3353      \bool_if:NF \l__enumext_store_active_bool
3354        {
3355          \msg_error:nnnn { enumext } { wrong-place }{ keyanspic }{ save-ans }
3356        }
3357      \int_compare:nNnT { \l__enumext_level_int } > { 1 }
3358        {
3359          \msg_error:nn { enumext } { keyanspic-wrong-level }
3360        }
3361      \int_compare:nNnT { \l__enumext_keyans_level_int } = { 1 }
3362        {
3363          \msg_error:nnnn { enumext } { command-wrong-place }{ anspic }{ keyans }
3364        }
```

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.

```
3365      \seq_put_right:Nn \l__enumext_keyans_pic_body_seq
3366        {
3367          \__enumext_keyans_anspic_code:nnn { #1 } { #2 } { #3 }
3368        }
3369    }
```

(*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.

```
3370    \cs_new_protected:Nn \__enumext_keyans_anspic_code:nnn
3371      {
3372        \stepcounter { enumXvi }
```

```
3373        #3 \\
3374        \bool_if:nT { #1 }
3375          {
3376            \__enumext_keyans_addto_prop:n { #2 }
3377            \__enumext_keyans_store_ref:
3378            \__enumext_keyans_addto_seq:n { #2 }
3379            \int_gincr:N \g__enumext_check_starred_cmd_int
3380            \bool_lazy_or:nnT
3381              { \bool_if_p:N \l__enumext_show_answer_bool }
3382              { \bool_if_p:N \l__enumext_show_position_bool }
3383              {
3384                \tl_set_eq:NN \l__enumext_label_v_tl \l__enumext_label_vi_tl
3385                \__enumext_keyans_show_left:n { #2 }
3386                \tl_set_eq:NN \l__enumext_label_vi_tl \l__enumext_label_v_tl
3387              }
3388          }
3389        \tl_use:N \l__enumext_label_font_style_v_tl
3390        \__enumext_wrapper_label_v:n { \l__enumext_label_vi_tl } \__enumext_keyans_show_item_opt:
3391      }
```

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

11.36.2 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.

```
3392  \NewDocumentEnvironment{keyanspic}{ o }
3393    {
3394      \__enumext_keyans_pic_safe_exec:
3395      \__enumext_start_list:nn
3396        { }
3397        {
3398          \__enumext_keyans_pic_arg_two:
3399        }
```

We apply the "adjusted" vertical spacing above the environment

```
3400        \vspace { \l__enumext_keyans_pic_above_skip }
3401    }
```

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.

```
3402    {
3403      \tl_if_novalue:nTF { #1 }
3404        {
3405          \__enumext_keyans_pic_do:e { \seq_count:N \l__enumext_keyans_pic_body_seq }
3406        }
3407        { \__enumext_keyans_pic_do:n { #1 } }
3408      \__enumext_stop_list:
3409      \__enumext_check_starred_cmd:n { anspic }
3410      \setcounter { enumXvi } { 0 }
3411      \vspace { \l__enumext_topsep_v_skip }
3412      %\bool_set_false:N \l__enumext_store_active_bool
3413    }
```

(*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.

```
3414  \cs_new_protected:Nn \__enumext_keyans_pic_safe_exec:
3415    {
3416      \int_incr:N \l__enumext_keyans_pic_level_int
3417      \int_compare:nNnT { \l__enumext_keyans_pic_level_int } > { 1 }
3418        {
3419          \msg_error:nn { enumext } { keyanspic-nested }
3420        }
3421      \__enumext_keyans_start_line:
3422    }
```

(*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.

```
3423  \cs_new_protected:Npn \__enumext_keyans_pic_skip_abs:N #1
3424    {
3425      \dim_compare:nNnT { #1 } < { 0pt }
3426        { \skip_set:Nn #1 { -#1 } }
3427    }
```

(*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.

```
3428  \cs_new_protected:Nn \__enumext_keyans_pic_arg_two:
3429    {
```

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.

```
3430      \bool_set_false:N \l__enumext_leftmargin_tmp_v_bool
3431      \__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.

```
3432      \skip_add:Nn \parsep { \itemsep }
3433      \dim_add:Nn  \leftmargin { -\labelwidth - \labelsep }
3434      \dim_zero:N  \labelwidth
3435      \dim_zero:N  \listparindent
3436      \dim_zero:N  \labelsep
3437      \skip_zero:N \partopsep
3438      \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.

```
3439      \__enumext_keyans_pic_skip_abs:N \parsep
3440      \skip_set:Nn \l__enumext_keyans_pic_above_skip
3441        {
3442          \box_dp:N \strutbox
3443          + \l__enumext_topsep_v_skip
3444          - \parsep
3445        }
3446      \__enumext_item_std:w \scan_stop:
3447    }
```

(*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.

```
3448  \cs_new_protected:Nn \__enumext_keyans_pic_do:n
3449    {
3450      \clist_map_function:nN { #1 } \__enumext_keyans_pic_row:n
3451    }
3452  \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.

```
3453  \cs_new_protected:Nn \__enumext_keyans_pic_row:n
3454    {
3455      \dim_set:Nn \l__enumext_keyans_pic_width_dim { \linewidth / #1 }
3456      \int_set:Nn \l__enumext_keyans_pic_above_int { \l__enumext_keyans_pic_below_int }
3457      \int_set:Nn \l__enumext_keyans_pic_below_int { \l__enumext_keyans_pic_above_int + #1 }
3458      \int_step_inline:nnn
3459        { \l__enumext_keyans_pic_above_int + 1 }
3460        { \l__enumext_keyans_pic_below_int }
3461        {
3462          \__enumext_minipage:w [ b ]{ \l__enumext_keyans_pic_width_dim }
3463            \centering
```

```
3464          \seq_item:Nn \l__enumext_keyans_pic_body_seq { ##1 }
3465        \__enumext_endminipage:
3466      }
3467    \par
3468  }
```

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

11.37 The environment enumext*

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

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

11.37.1 Functions for item box width

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

```
3469 \cs_new_protected:Nn \__enumext_starred_columns_set_vii:
3470  {
3471    \dim_compare:nNnT { \l__enumext_columns_sep_vii_dim } = { \c_zero_dim }
3472      {
3473        \dim_set:Nn \l__enumext_columns_sep_vii_dim
3474          {
3475            ( \l__enumext_labelwidth_vii_dim + \l__enumext_labelsep_vii_dim )
3476            / \l__enumext_columns_vii_int
3477          }
3478      }
3479    \int_set:Nn \l__enumext_tmpa_vii_int { \l__enumext_columns_vii_int - \c_one_int }
3480    \dim_set:Nn \l__enumext_item_width_vii_dim
3481      {
3482        ( \linewidth - \l__enumext_columns_sep_vii_dim * \l__enumext_tmpa_vii_int )
3483        / \l__enumext_columns_vii_int - \l__enumext_labelwidth_vii_dim
3484        - \l__enumext_labelsep_vii_dim
3485      }
3486    \dim_zero_new:N \itemwidth
3487  }
```

(*End of definition for* __enumext_starred_columns_set_vii:.)

__enumext_starred_joined_item_vii:n   The function `\__enumext_starred_joined_item_vii:n` will set the *width* of the box in which the content passed to `\item`(⟨*number*⟩) will be stored together with the value of `\itemwidth`.

```
3488 \cs_new_protected:Npn \__enumext_starred_joined_item_vii:n #1
3489  {
3490    \int_set:Nn \l__enumext_joined_item_vii_int {#1}
3491    \int_compare:nNnT { \l__enumext_joined_item_vii_int } > { \l__enumext_columns_vii_int }
3492      {
3493        \msg_warning:nnee { enumext } { item-joined }
3494          { \int_use:N \l__enumext_joined_item_vii_int }
3495          { \int_use:N \l__enumext_columns_vii_int }
3496        \int_set:Nn \l__enumext_joined_item_vii_int
3497          {
3498            \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + \c_one_int
3499          }
3500      }
3501    \int_compare:nNnT
3502      { \l__enumext_joined_item_vii_int }
3503        >
3504      { \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + \c_one_int }
3505      {
3506        \msg_warning:nnee { enumext } { item-joined-columns }
3507          { \int_use:N \l__enumext_joined_item_vii_int }
3508          {
3509            \int_eval:n
3510              { \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + \c_one_int }
```

```
3511              }
3512          \int_set:Nn \l__enumext_joined_item_vii_int
3513            {
3514              \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + \c_one_int
3515            }
3516        }
```

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

```
3517        \int_compare:nNnTF { \l__enumext_joined_item_vii_int } > { \c_one_int }
3518          {
3519            \int_set_eq:NN \l__enumext_joined_item_aux_vii_int \l__enumext_joined_item_vii_int
3520            \int_decr:N \l__enumext_joined_item_aux_vii_int
3521            \int_add:Nn \l__enumext_item_column_pos_vii_int { \l__enumext_joined_item_aux_vii_int }
3522            \int_gadd:Nn \g__enumext_item_count_all_vii_int { \l__enumext_joined_item_aux_vii_int }
3523            \dim_set:Nn \l__enumext_joined_width_vii_dim
3524              {
3525                \l__enumext_item_width_vii_dim * \l__enumext_joined_item_vii_int
3526              + (  \l__enumext_labelwidth_vii_dim + \l__enumext_labelsep_vii_dim
3527                + \l__enumext_columns_sep_vii_dim
3528              )*\l__enumext_joined_item_aux_vii_int
3529              }
3530            \dim_set_eq:NN \itemwidth \l__enumext_joined_width_vii_dim
3531          }
3532          {
3533            \dim_set_eq:NN \l__enumext_joined_width_vii_dim \l__enumext_item_width_vii_dim
3534            \dim_set_eq:NN \itemwidth \l__enumext_item_width_vii_dim
3535          }
3536      }
```

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

__enumext_start_mini_vii: The implementation of the `mini-env` key support is almost identical to the one used in the `enumext` and `keyans` environments, the difference is that the `__enumext_mini_env*` environment on the *"right side"* is executed *"after"* closing the environment, so it is necessary to make a global copy of the variable \l__enumext_minipage_right_vii_dim in the variable \g__enumext_minipage_right_vii_dim.

```
3537  \cs_new_protected:Nn \__enumext_start_mini_vii:
3538    {
3539      \dim_compare:nNnT { \l__enumext_minipage_right_vii_dim } > { \c_zero_dim }
3540        {
3541          \dim_set:Nn \l__enumext_minipage_left_vii_dim
3542            {
3543              \linewidth
3544              - \l__enumext_minipage_right_vii_dim
3545              - \l__enumext_minipage_hsep_vii_dim
3546            }
3547          \bool_set_true:N \l__enumext_minipage_active_vii_bool
3548          \dim_gset_eq:NN
3549            \g__enumext_minipage_right_vii_dim
3550            \l__enumext_minipage_right_vii_dim
3551          \__enumext_mini_addvspace_vii:
3552          \nointerlineskip\noindent
3553          \begin{__enumext_mini_env*}{ \l__enumext_minipage_left_vii_dim }
3554        }
3555    }
```

(*End of definition for __enumext_start_mini_vii:.*)

__enumext_stop_mini_vii: The function __enumext_stop_mini_vii: closes the `__enumext_mini_env*` environment on the left side, applies \hfill and sets the value of the variable \g__enumext_minipage_active_vii_bool to true which will be used in the function __enumext_after_star_env:nn to execute the `__enumext_-mini_env*` on the *"right side"*.

```
3556  \cs_new_protected:Nn \__enumext_stop_mini_vii:
3557    {
3558      \bool_if:NT \l__enumext_minipage_active_vii_bool
3559        {
3560          \end{__enumext_mini_env*}
3561          \hfill
3562          \bool_gset_true:N \g__enumext_minipage_active_vii_bool
3563        }
3564    }
```

Finally we execute 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"*.

```
3565  \__enumext_after_env:nn {enumext*}
3566    {
3567      \bool_if:NT \g__enumext_minipage_active_vii_bool
3568        {
3569          \begin{__enumext_mini_env*}{ \g__enumext_minipage_right_vii_dim }
3570            \par\addvspace { \g__enumext_minipage_right_skip }
3571            \bool_if:NF \g__enumext_minipage_center_vii_bool
3572              {
3573                \centering
3574              }
3575            \tl_use:N \g__enumext_miniright_code_vii_tl % the code
3576          \end{__enumext_mini_env*}
3577          \par\addvspace{ \g__enumext_minipage_after_skip }
3578        }
3579      \bool_gset_false:N \g__enumext_minipage_active_vii_bool
3580      \bool_gset_true:N \g__enumext_minipage_center_vii_bool
3581      \tl_gclear:N \g__enumext_miniright_code_vii_tl
3582      \dim_gzero:N \g__enumext_minipage_right_vii_dim
3583      \bool_gset_false:N \g__enumext_starred_bool
3584    }
```

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

enumext*   First we will generate the environment and we will give a temporary definition to `\__enumext_stop_-item_tmp_vii:` equal to `\noindent` and next to `\item` equal to `\__enumext_start_item_tmp_vii:` which we will redefine later.

```
3585  \NewDocumentEnvironment{enumext*}{ o }
3586    {
3587      \__enumext_safe_exec_vii:
3588      \__enumext_parse_keys_vii:n {#1}
3589      \__enumext_before_list_vii:
3590      \__enumext_start_store_level_vii:
3591      \__enumext_start_list:nn { }
3592        {
3593          \__enumext_list_arg_two_vii:
3594          \__enumext_before_keys_exec_vii:
3595        }
3596      \__enumext_starred_columns_set_vii:
3597      \item[] \scan_stop:
3598      \cs_set_eq:NN \__enumext_stop_item_tmp_vii: \noindent
3599      \cs_set_eq:NN \item \__enumext_start_item_tmp_vii:
3600    }
3601    {
3602      \__enumext_stop_item_tmp_vii:
3603      \__enumext_remove_extra_parsep_vii:
3604      \__enumext_stop_list:
3605      \__enumext_stop_store_level_vii:
3606      \__enumext_after_list_vii:
3607    }
```

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

`\__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`.

```
3608  \cs_new_protected:Nn \__enumext_safe_exec_vii:
3609    {
3610      \__enumext_internal_mini_page:
3611      \__enumext_is_not_nested:
3612      \int_incr:N \l__enumext_level_h_int
3613      \int_compare:nNnT { \l__enumext_level_h_int } > { 1 }
3614        {
3615          \msg_error:nn { enumext } { nested }
3616        }
3617      \bool_set_true:N \l__enumext_starred_bool
3618      \__enumext_is_on_first_level:
3619    }
```

(*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⟩.

```
3620 \cs_new_protected:Npn \__enumext_parse_keys_vii:n #1
3621   {
3622     \tl_if_novalue:nF {#1}
3623       {
3624         \str_clear:N \l__enumext_series_str
3625         \keys_set:nn { enumext / enumext* } {#1}
3626         \__enumext_parse_series:n {#1}
3627         \__enumext_store_active_keys_vii:n {#1}
3628       }
3629   }
```

(*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.

```
3630 \cs_new_protected:Nn \__enumext_before_list_vii:
3631   {
3632     \__enumext_vspace_above_vii:
3633     \__enumext_check_ans_active:
3634     \__enumext_before_args_exec_vii:
3635     \__enumext_start_mini_vii:
3636   }
```

(*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.

```
3637 \cs_new_protected:Nn \__enumext_after_list_vii:
3638   {
3639     \__enumext_stop_mini_vii:
3640     \__enumext_after_stop_list_vii:
3641     \__enumext_check_ans_key_hook:
3642     \__enumext_vspace_below_vii:
3643     \bool_set_false:N \l__enumext_starred_bool
3644     \__enumext_resume_save_counter:
3645   }
```

(*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.

```
3646 \cs_new_protected:Nn \__enumext_start_store_level_vii:
3647   {
3648     \bool_if:NT \l__enumext_store_active_bool
3649       {
3650         \int_compare:nNnT { \l__enumext_level_int } > { \c_zero_int }
3651           {
3652             \__enumext_store_level_open_vii:
3653           }
3654       }
3655   }
3656 \cs_new_protected:Nn \__enumext_stop_store_level_vii:
3657   {
3658     \bool_if:NT \l__enumext_store_active_bool
3659       {
3660         \int_compare:nNnT { \l__enumext_level_int } > { \c_zero_int }
3661           {
3662             \__enumext_store_level_close_vii:
3663           }
3664       }
3665   }
```

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

11.37.2 The command \item in enumext*

__enumext_start_item_tmp_vii:

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

```
3666 \cs_new_protected_nopar:Nn \__enumext_start_item_tmp_vii:
3667   {
3668     \__enumext_stop_item_tmp_vii:
3669     \int_incr:N \l__enumext_item_column_pos_vii_int
3670     \int_gincr:N \g__enumext_item_count_all_vii_int
3671     \__enumext_item_peek_args_vii:
3672   }
```

(*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).

```
3673 \cs_new_protected:Nn \__enumext_item_peek_args_vii:
3674   {
3675     \peek_meaning:NTF (
3676       { \__enumext_joined_item_vii:w }
3677       { \__enumext_joined_item_vii:w (1) }
3678   }
```

(*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.

```
3679 \cs_new_protected:Npn \__enumext_joined_item_vii:w (#1)
3680   {
3681     \__enumext_starred_joined_item_vii:n {#1}
3682     \peek_meaning_remove:NTF *
3683       { \__enumext_starred_item_vii:w  }
3684       { \__enumext_standar_item_vii:w }
3685   }
```

(*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].

```
3686 \cs_new_protected:Npn \__enumext_standar_item_vii:w
3687   {
3688     \bool_set_false:N \l__enumext_item_starred_vii_bool
3689       \peek_meaning:NTF [
3690         {
3691           \bool_set_eq:NN
3692             \l__enumext_wrap_label_vii_bool
3693             \l__enumext_wrap_label_opt_vii_bool
3694           \__enumext_start_item_vii:w
3695         }
3696         {
3697           \bool_set_true:N \l__enumext_wrap_label_vii_bool
3698           \legacy_if_set_true:n { @noitemarg }
3699           \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
3700         }
3701   }
```

(*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*⟩`]`.

```
3702 \cs_new_protected:Npn \__enumext_starred_item_vii:w
3703   {
3704     \bool_set_true:N \l_enumext_item_starred_vii_bool
3705     \bool_set_true:N \l_enumext_wrap_label_vii_bool
3706     \peek_meaning:NTF [
3707       { \__enumext_starred_item_vii_aux_i:w }
3708       { \__enumext_starred_item_vii_aux_ii:w }
3709   }
3710 \cs_new_protected:Npn \__enumext_starred_item_vii_aux_i:w [#1]
3711   {
3712     \tl_gset:Nn \g__enumext_item_symbol_aux_vii_tl {#1}
3713     \__enumext_starred_item_vii_aux_ii:w
3714   }
3715 \cs_new_protected:Npn \__enumext_starred_item_vii_aux_ii:w
3716   {
3717     \peek_meaning:NTF [
3718       { \__enumext_starred_item_vii_aux_iii:w }
3719       {
3720         \dim_set_eq:NN
3721           \l__enumext_item_symbol_sep_vii_dim
3722           \l__enumext_labelsep_vii_dim
3723         \legacy_if_set_true:n { @noitemarg }
3724         \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
3725       }
3726   }
3727 \cs_new_protected:Npn \__enumext_starred_item_vii_aux_iii:w [#1]
3728   {
3729     \dim_set:Nn \l__enumext_item_symbol_sep_vii_dim {#1}
3730     \legacy_if_set_true:n { @noitemarg }
3731     \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
3732   }
```

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

11.37.3   Real definition of `\item` in enumext*

`\__enumext_start_item_vii:w`

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

The first thing we will do is set the value of `\__enumext_stop_item_tmp_vii:` equal to `\__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.

```
3733 \cs_new_protected_nopar:Npn \__enumext_start_item_vii:w [#1]
3734   {
3735     \cs_set_eq:NN \__enumext_stop_item_tmp_vii: \__enumext_stop_item_vii:
3736     \legacy_if:nT { @noitemarg }
3737       {
3738         \legacy_if_set_false:n { @noitemarg }
3739         \legacy_if:nT { @nmbrlist }
3740           {
3741             \bool_if:NT \l__enumext_hyperref_bool
3742               {
3743                 \legacy_if_set_true:n { @hyper@item }
3744               }
3745             \refstepcounter{enumXvii}
3746             \bool_if:NT \l__enumext_check_answers_bool
3747               {
3748                 \int_gincr:N \g__enumext_item_number_int
3749               }
3750           }
3751       }
```

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.

```
3752        \group_begin:
3753          \lrbox{ \l__enumext_item_text_vii_box }
3754            \bool_if:NF \l__enumext_footnotes_key_bool
3755              {
3756                \__enumext_renew_footnote:
3757              }
3758            \bool_if:NT \l__enumext_item_starred_vii_bool
3759              {
3760                \tl_if_blank:VT \g__enumext_item_symbol_aux_vii_tl
3761                  {
3762                    \tl_gset_eq:NN
3763                      \g__enumext_item_symbol_aux_vii_tl \l__enumext_item_symbol_vii_tl
3764                  }
3765                \mode_leave_vertical:
3766                \skip_horizontal:n { -\l__enumext_item_symbol_sep_vii_dim }
3767                \makebox[ 0pt ][ r ]{ \g__enumext_item_symbol_aux_vii_tl }
3768                \skip_horizontal:N \l__enumext_item_symbol_sep_vii_dim
3769                \tl_gclear:N \g__enumext_item_symbol_aux_vii_tl
3770              }
3771            \group_begin:
3772              \tl_use:N \l__enumext_label_font_style_vii_tl
3773              \bool_if:NTF \l__enumext_wrap_label_vii_bool
3774                {
3775                  \makebox[ \l__enumext_labelwidth_vii_dim ][ \l__enumext_align_label_vii_str ]
3776                    { \__enumext_wrapper_label_vii:n {#1} }
3777                }
3778                {
3779                  \makebox[ \l__enumext_labelwidth_vii_dim ][ \l__enumext_align_label_vii_str ]{ #1 }
3780                }
3781            \group_end:
3782            \skip_horizontal:N \l__enumext_labelsep_vii_dim
3783            \tl_use:N \l__enumext_after_list_args_vii_tl
3784            \__enumext_minipage:w [ t ]{ \l__enumext_joined_width_vii_dim  }
3785              \skip_set_eq:NN \parindent \l__enumext_listparindent_vii_dim
3786              \skip_set_eq:NN \parskip \l__enumext_parsep_vii_skip
3787              \tl_use:N \l__enumext_fake_item_indent_vii_tl
3788        }
```

(*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.

```
3789  \cs_new_protected_nopar:Nn \__enumext_stop_item_vii:
3790    {
3791        \__enumext_endminipage:
3792      \endlrbox
3793    \group_end:
3794    \box_set_wd:Nn \l__enumext_item_text_vii_box
3795      {
3796        \l__enumext_joined_width_vii_dim
3797        + \l__enumext_labelwidth_vii_dim
3798        + \l__enumext_labelsep_vii_dim
3799      }
3800    \int_set:Nn \hbadness { 10000 }
3801    \box_use:N \l__enumext_item_text_vii_box
3802    \bool_if:NF \l__enumext_footnotes_key_bool
3803      {
3804        \__enumext_print_footnote:
3805      }
3806    \int_compare:nNnTF { \l__enumext_item_column_pos_vii_int } = { \l__enumext_columns_vii_int }
3807      {
3808        \par\noindent
3809        \int_zero:N \l__enumext_item_column_pos_vii_int
3810      }
3811      { \hspace{ \l__enumext_columns_sep_vii_dim } }
3812    }
```

(*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.

```
3813 \cs_new_protected:Nn \__enumext_remove_extra_parsep_vii:
3814   {
3815     \int_compare:nNnT
3816       {
3817         \int_mod:nn {  \g__enumext_item_count_all_vii_int } { \l__enumext_columns_vii_int }
3818       }
3819       =
3820       { \c_zero_int }
3821       {
3822         \par
3823         \vspace{ -\l__enumext_itemsep_vii_skip }
3824         \int_gzero:N \g__enumext_item_count_all_vii_int
3825       }
3826   }
```

(*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.

```
3827 \__enumext_after_env:nn {enumext*} { \__enumext_execute_after_env: }
```

11.38   The environment keyans*

11.38.1   Functions for item box width

__enumext_starred_columns_set_viii:

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

```
3828 \cs_new_protected:Nn \__enumext_starred_columns_set_viii:
3829   {
3830     \dim_compare:nNnT { \l__enumext_columns_sep_viii_dim } = { \c_zero_dim }
3831       {
3832         \dim_set:Nn \l__enumext_columns_sep_viii_dim
3833           {
3834             ( \l__enumext_labelwidth_viii_dim + \l__enumext_labelsep_viii_dim )
3835             / \l__enumext_columns_viii_int
3836           }
3837       }
3838     \int_set:Nn \l__enumext_tmpa_viii_int { \l__enumext_columns_viii_int - \c_one_int }
3839     \dim_set:Nn \l__enumext_item_width_viii_dim
3840       {
3841         ( \linewidth - \l__enumext_columns_sep_viii_dim * \l__enumext_tmpa_viii_int )
3842         / \l__enumext_columns_viii_int - \l__enumext_labelwidth_viii_dim
3843         - \l__enumext_labelsep_viii_dim
3844       }
3845     \dim_zero_new:N \itemwidth
3846   }
```

(*End of definition for* __enumext_starred_columns_set_viii:.)

__enumext_starred_joined_item_viii:n

The function __enumext_starred_joined_item_viii:n will set the *width* of the box in which the content passed to \item(⟨*number*⟩) will be stored together with the value of \itemwidth.

```
3847 \cs_new_protected:Npn \__enumext_starred_joined_item_viii:n #1
3848   {
3849     \int_set:Nn \l__enumext_joined_item_viii_int {#1}
3850     \int_compare:nNnT { \l__enumext_joined_item_viii_int } > { \l__enumext_columns_viii_int }
3851       {
3852         \msg_warning:nnee { enumext } { item-joined }
3853           { \int_use:N \l__enumext_joined_item_viii_int }
3854           { \int_use:N \l__enumext_columns_viii_int }
3855         \int_set:Nn \l__enumext_joined_item_viii_int
3856           {
3857             \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + \c_one_int
3858           }
3859       }
3860     \int_compare:nNnT
3861       { \l__enumext_joined_item_viii_int }
3862       >
3863       { \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + \c_one_int }
3864       {
3865         \msg_warning:nnee { enumext } { item-joined-columns }
```

```
3866            { \int_use:N \l__enumext_joined_item_viii_int }
3867            {
3868              \int_eval:n
3869                { \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + \c_one_int
3870            }
3871          \int_set:Nn \l__enumext_joined_item_viii_int
3872            {
3873              \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + \c_one_int
3874            }
3875        }
3876      \int_compare:nNnTF { \l__enumext_joined_item_viii_int } > { \c_one_int }
3877        {
3878          \int_set_eq:NN \l__enumext_joined_item_aux_viii_int \l__enumext_joined_item_viii_int
3879          \int_decr:N \l__enumext_joined_item_aux_viii_int
3880          \int_add:Nn \l__enumext_item_column_pos_viii_int { \l__enumext_joined_item_aux_viii_int
3881          \int_gadd:Nn \g__enumext_item_count_all_viii_int { \l__enumext_joined_item_aux_viii_int
3882          \dim_set:Nn \l__enumext_joined_width_viii_dim
3883            {
3884              \l__enumext_item_width_viii_dim * \l__enumext_joined_item_viii_int
3885              + (  \l__enumext_labelwidth_viii_dim + \l__enumext_labelsep_viii_dim
3886                + \l__enumext_columns_sep_viii_dim
3887              )*\l__enumext_joined_item_aux_viii_int
3888            }
3889          \dim_set_eq:NN \itemwidth \l__enumext_joined_width_viii_dim
3890        }
3891        {
3892          \dim_set_eq:NN \l__enumext_joined_width_viii_dim \l__enumext_item_width_viii_dim
3893          \dim_set_eq:NN \itemwidth \l__enumext_item_width_viii_dim
3894        }
3895    }
```

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

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

```
3896  \cs_new_protected:Nn \__enumext_start_mini_viii:
3897    {
3898      \dim_compare:nNnT { \l__enumext_minipage_right_viii_dim } > { \c_zero_dim }
3899        {
3900          \dim_set:Nn \l__enumext_minipage_left_viii_dim
3901            {
3902              \linewidth
3903              - \l__enumext_minipage_right_viii_dim
3904              - \l__enumext_minipage_hsep_viii_dim
3905            }
3906          \bool_set_true:N \l__enumext_minipage_active_viii_bool
3907          \dim_gset_eq:NN
3908            \g__enumext_minipage_right_viii_dim
3909            \l__enumext_minipage_right_viii_dim
3910          \__enumext_mini_addvspace_viii:
3911          \nointerlineskip\noindent
3912          \begin{__enumext_mini_env*}{ \l__enumext_minipage_left_viii_dim }
3913        }
3914    }
3915  \cs_new_protected:Nn \__enumext_stop_mini_viii:
3916    {
3917      \bool_if:NT \l__enumext_minipage_active_viii_bool
3918        {
3919          \end{__enumext_mini_env*}
3920          \hfill
3921          \bool_gset_true:N \g__enumext_minipage_active_viii_bool
3922        }
3923    }
3924  \__enumext_after_env:nn {keyans*}
3925    {
3926      \bool_if:NT \g__enumext_minipage_active_viii_bool
3927        {
3928          \begin{__enumext_mini_env*}{ \g__enumext_minipage_right_viii_dim }
3929            \par\addvspace { \g__enumext_minipage_right_skip }
3930            \bool_if:NF \g__enumext_minipage_center_viii_bool
3931              {
3932                \centering
```

```
3933                    }
3934                \tl_use:N \g__enumext_miniright_code_viii_tl % the code
3935            \end{__enumext_mini_env*}
3936            \par\addvspace{ \g__enumext_minipage_after_skip }
3937        }
3938    \bool_gset_false:N \g__enumext_minipage_active_viii_bool
3939    \bool_gset_true:N \g__enumext_minipage_center_viii_bool
3940    \tl_gclear:N \g__enumext_miniright_code_viii_tl
3941    \dim_gzero:N \g__enumext_minipage_right_viii_dim
3942    }
```

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

keyans* First we will generate the environment and we will give a temporary definition to __enumext_stop_-item_tmp_viii: equal to \noindent and next to \item equal to __enumext_start_item_tmp_-viii: which we will redefine later.

```
3943    \NewDocumentEnvironment{keyans*}{ o }
3944    {
3945        \__enumext_safe_exec_viii:
3946        \__enumext_parse_keys_viii:n {#1}
3947        \__enumext_before_list_viii:
3948        \__enumext_start_list:nn { }
3949        {
3950            \__enumext_list_arg_two_viii:
3951            \__enumext_before_keys_exec_viii:
3952        }
3953        \__enumext_starred_columns_set_viii:
3954        \item[] \scan_stop:
3955        \cs_set_eq:NN \__enumext_stop_item_tmp_viii: \noindent
3956        \cs_set_eq:NN \item \__enumext_start_item_tmp_viii:
3957    }
3958    {
3959        \__enumext_stop_item_tmp_viii:
3960        \__enumext_remove_extra_parsep_viii:
3961        \__enumext_check_starred_cmd:n { item }
3962        \__enumext_stop_list:
3963        \__enumext_after_list_viii:
3964    }
```

(*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.

```
3965    \cs_new_protected:Nn \__enumext_safe_exec_viii:
3966    {
3967        \int_incr:N \l__enumext_keyans_level_h_int
3968        \int_compare:nNnT { \l__enumext_keyans_level_h_int } > { 1 }
3969        {
3970            \msg_error:nn { enumext } { nested }
3971        }
3972        \__enumext_keyans_start_line:
3973        % Set false for interfering with enumext nested in keyans* (yes, its possible and crayze)
3974        \bool_set_false:N \l__enumext_store_active_bool
3975        \int_compare:nNnT { \l__enumext_level_int } > { 1 }
3976        {
3977            \msg_error:nn { enumext } { keyans-wrong-level }
3978        }
3979    }
```

(*End of definition for* __enumext_safe_exec_viii:.)

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

```
3980    \cs_new_protected:Npn \__enumext_parse_keys_viii:n #1
3981    {
3982        \tl_if_novalue:nF {#1}
3983        {
3984            \keys_set:nn { enumext / keyans* } {#1}
3985        }
3986    }
```

(*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.

```
3987  \cs_new_protected:Nn \__enumext_before_list_viii:
3988    {
3989      \__enumext_vspace_above_viii:
3990      \__enumext_before_args_exec_viii:
3991      \__enumext_start_mini_viii:
3992    }
```

(*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.

```
3993  \cs_new_protected:Nn \__enumext_after_list_viii:
3994    {
3995      \__enumext_stop_mini_viii:
3996      \__enumext_after_stop_list_viii:
3997      \__enumext_vspace_below_viii:
3998    }
```

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

11.38.2 The command \item in keyans*

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

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

```
3999  \cs_new_protected_nopar:Nn \__enumext_start_item_tmp_viii:
4000    {
4001      \__enumext_stop_item_tmp_viii:
4002      \int_incr:N \l__enumext_item_column_pos_viii_int
4003      \int_gincr:N \g__enumext_item_count_all_viii_int
4004      \__enumext_item_peek_args_viii:
4005    }
```

(*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).

```
4006  \cs_new_protected:Nn \__enumext_item_peek_args_viii:
4007    {
4008      \peek_meaning:NTF (
4009        { \__enumext_joined_item_viii:w }
4010        { \__enumext_joined_item_viii:w (1) }
4011    }
```

(*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`.

```
4012  \cs_new_protected:Npn \__enumext_joined_item_viii:w (#1)
4013    {
4014      \__enumext_starred_joined_item_viii:n {#1}
4015      \peek_meaning_remove:NTF *
4016        { \__enumext_starred_item_viii:w  }
4017        { \__enumext_standar_item_viii:w }
4018    }
```

(*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 ]`.

```
4019  \cs_new_protected:Npn \__enumext_standar_item_viii:w
4020    {
4021      \bool_set_false:N \l__enumext_item_starred_viii_bool
4022        \peek_meaning:NTF [
4023          {
4024            \bool_set_eq:NN
4025              \l__enumext_wrap_label_viii_bool
4026              \l__enumext_wrap_label_opt_viii_bool
4027            \__enumext_start_item_viii:w
4028          }
4029          {
4030            \bool_set_true:N \l__enumext_wrap_label_viii_bool
4031            \legacy_if_set_true:n { @noitemarg }
4032            \__enumext_start_item_viii:w [ \l__enumext_label_viii_tl ]
4033          }
4034    }
```

(*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*⟩`]`.

```
4035  \cs_new_protected:Npn \__enumext_starred_item_viii:w
4036    {
4037      \bool_set_true:N \l__enumext_item_starred_viii_bool
4038      \bool_set_true:N \l__enumext_wrap_label_viii_bool
4039      \peek_meaning:NTF [
4040        { \__enumext_starred_item_viii_aux_i:w }
4041        { \__enumext_starred_item_viii_aux_ii:w }
4042    }
```

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`.

```
4043  \cs_new_protected:Npn \__enumext_starred_item_viii_aux_i:w [#1]
4044    {
4045      \tl_clear:N \l__enumext_store_current_label_tl
4046      \tl_if_novalue:nF { #1 }
4047        {
4048          \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_tl
4049            {
4050              \tl_put_right:Ne \l__enumext_store_current_label_tl { \l__enumext_store_keyans_item_o|
4051              \tl_put_right:Ne \l__enumext_store_current_label_tl { #1 }
4052            }
4053          \tl_set:Ne \l__enumext_store_current_opt_arg_tl { #1 }
4054        }
4055      \__enumext_starred_item_viii_aux_ii:w
4056    }
4057  \cs_new_protected:Npn \__enumext_starred_item_viii_aux_ii:w
4058    {
4059      \legacy_if_set_true:n { @noitemarg }
4060      \__enumext_start_item_viii:w [ \l__enumext_label_viii_tl ]
4061    }
```

(*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.

```
4062  \cs_new_protected:Nn \__enumext_starred_item_exec:
4063    {
4064      \tl_put_left:Ne \l__enumext_store_current_label_tl { \l__enumext_label_viii_tl }
4065      \__enumext_store_addto_prop:V \l__enumext_store_current_label_tl
4066      \__enumext_keyans_store_ref:
4067      \tl_put_left:Ne \l__enumext_store_current_label_tl { \item }
4068      \__enumext_keyans_addto_seq_link:
4069      \int_gincr:N \g__enumext_check_starred_cmd_int
4070      \bool_if:NT \l__enumext_show_answer_bool
4071        {
4072          \__enumext_print_keyans_box:NN \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
4073        }
4074      \bool_if:NT \l__enumext_show_position_bool
4075        {
4076          \tl_set:Ne \l__enumext_mark_answer_sym_tl
4077            {
4078              \group_begin:
4079                \exp_not:N \normalfont
4080                \exp_not:N \footnotesize [ \int_eval:n
4081                  {
4082                    \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
4083                  }
4084                ]
4085              \group_end:
4086            }
4087          \__enumext_print_keyans_box:NN \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
4088        }
4089    }
```

(*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.

```
4090  \cs_new_protected_nopar:Npn \__enumext_start_item_viii:w [#1]
4091    {
4092      \cs_set_eq:NN \__enumext_stop_item_tmp_viii: \__enumext_stop_item_viii:
4093      \legacy_if:nT { @noitemarg }
4094        {
4095          \legacy_if_set_false:n { @noitemarg }
4096          \legacy_if:nT { @nmbrlist }
4097            {
4098              \bool_if:NT \l__enumext_hyperref_bool
4099                {
4100                  \legacy_if_set_true:n { @hyper@item }
4101                }
4102              \refstepcounter{enumXviii}
4103            }
4104        }
```

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

```
4105      \group_begin:
4106        \lrbox{ \l__enumext_item_text_viii_box }
4107          \bool_if:NF \l__enumext_footnotes_key_bool
4108            {
4109              \__enumext_renew_footnote:
4110            }
4111          \bool_if:NT \l__enumext_item_starred_viii_bool
4112            {
4113              \__enumext_starred_item_exec:
4114            }
4115          \group_begin:
4116            \tl_use:N \l__enumext_label_font_style_viii_tl
4117            \bool_if:NTF \l__enumext_wrap_label_viii_bool
4118              {
4119                \makebox[ \l__enumext_labelwidth_viii_dim ][ \l__enumext_align_label_viii_str ]
4120                  { \__enumext_wrapper_label_viii:n {#1} }
4121              }
4122              {
4123                \makebox[ \l__enumext_labelwidth_viii_dim ][ \l__enumext_align_label_viii_str ]{ #1
```

```
4124                }
4125            \group_end:
4126            \skip_horizontal:N \l__enumext_labelsep_viii_dim
4127            \tl_use:N \l__enumext_after_list_args_viii_tl
4128            \__enumext_minipage:w [ t ]{ \l__enumext_joined_width_viii_dim  }
4129              \skip_set_eq:NN \parindent \l__enumext_listparindent_viii_dim
4130              \skip_set_eq:NN \parskip \l__enumext_parsep_viii_skip
4131              \bool_if:NT \l__enumext_item_starred_viii_bool
4132                {
4133                  \tl_use:N \l__enumext_fake_item_indent_viii_tl
4134                  \__enumext_keyans_show_item_opt:
4135                  \skip_horizontal:n { -\l__enumext_fake_item_indent_viii_dim - \l__enumext_labelsep_
4136                }
4137                {
4138                  \tl_use:N \l__enumext_fake_item_indent_viii_tl
4139                }
4140        }
```

(*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.

```
4141  \cs_new_protected_nopar:Nn \__enumext_stop_item_viii:
4142    {
4143            \__enumext_endminipage:
4144          \endlrbox
4145        \group_end:
4146        \box_set_wd:Nn \l__enumext_item_text_viii_box
4147          {
4148            \l__enumext_joined_width_viii_dim
4149            + \l__enumext_labelwidth_viii_dim
4150            + \l__enumext_labelsep_viii_dim
4151          }
4152        \int_set:Nn \hbadness { 10000 }
4153        \box_use:N \l__enumext_item_text_viii_box
4154        \bool_if:NF \l__enumext_footnotes_key_bool
4155          {
4156            \__enumext_print_footnote:
4157          }
4158        \int_compare:nNnTF
4159          { \l__enumext_item_column_pos_viii_int } = { \l__enumext_columns_viii_int }
4160          {
4161            \par\noindent
4162            \int_zero:N \l__enumext_item_column_pos_viii_int
4163          }
4164          { \hspace{ \l__enumext_columns_sep_viii_dim } }
4165      }
```

(*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.

```
4166  \cs_new_protected:Nn \__enumext_remove_extra_parsep_viii:
4167    {
4168      \int_compare:nNnT
4169        {
4170          \int_mod:nn
4171            { \g__enumext_item_count_all_viii_int }
4172            { \l__enumext_columns_viii_int }
4173        }
4174        =
4175        { \c_zero_int }
4176        {
4177          \par
4178          \vspace{ -\l__enumext_itemsep_viii_skip }
4179          \int_gzero:N \g__enumext_item_count_all_viii_int
4180        }
4181    }
```

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

11.39   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.

```
4182  \NewDocumentCommand \getkeyans { m }
4183    {
4184      \exp_args:Ne \__enumext_getkeyans_aux:n
4185        { \tl_to_str:e { \text_expand:n {#1} } }
4186    }
```

(*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.

```
4187  \cs_new_protected:Npn \__enumext_getkeyans_aux:n #1
4188    {
4189      \str_if_in:nnTF {#1} { : }
4190        {
4191          \use:e
4192            {
4193              \cs_set:Npn \exp_not:N \__enumext_tmp:w ##1 \c_colon_str ##2 \scan_stop:
4194                { {##1} {##2} }
4195            }
4196          \exp_after:wN \__enumext_getkeyans:nn \__enumext_tmp:w #1 \scan_stop:
4197        }
4198        { \msg_error:nnn { enumext } { missing-colon } {#1} }
4199    }
```

(*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*⟩.

```
4200  \cs_new_protected:Npn \__enumext_getkeyans:nn #1 #2
4201    {
4202      \prop_if_exist:cF { g__enumext_#1_prop }
4203        { \msg_error:nnn { enumext } { undefined-storage-anskey } {#1} }
4204      \group_begin:
4205        \prop_item:cn { g__enumext_#1_prop }{#2}
4206      \group_end:
4207    }
```

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

11.40   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*⟩]

```
4208  \cs_generate_variant:Nn \keys_precompile:nnN { neN }
4209  \keys_define:nn  { enumext / print }
4210    {
4211    print*  .code:n    = \keys_precompile:neN { enumext / enumext* }
4212                          { \__enumext_filter_save_key:n {#1} }
4213                          \l__enumext_print_keyans_starred_tl, % starred cmd
4214    print*  .initial:n = { nosep, label=\arabic*., columns=2, first=\small, font=\small },
4215    print-1 .code:n    = \keys_precompile:neN { enumext / level-1 }
4216                          { \__enumext_filter_save_key:n {#1} }
4217                          \l__enumext_print_keyans_i_tl,
4218    print-1 .initial:n = { nosep, label=\arabic*., columns=2, first=\small, font=\small },
4219    print-2 .code:n    = \keys_precompile:neN { enumext / level-2 }
4220                          { \__enumext_filter_save_key:n {#1} }
4221                          \l__enumext_print_keyans_ii_tl,
4222    print-2 .initial:n = { nosep, label=(\alph*), first=\small, font=\small },
4223    print-3 .code:n    = \keys_precompile:neN { enumext / level-3 }
```

```
4224                              { \__enumext_filter_save_key:n {#1} }
4225                              \l__enumext_print_keyans_iii_tl,
4226      print-3 .initial:n  = { nosep, label=\roman*., first=\small, font=\small },
4227      print-4 .code:n     = \keys_precompile:neN { enumext / level-4 }
4228                              { \__enumext_filter_save_key:n {#1} }
4229                              \l__enumext_print_keyans_iv_tl,
4230      print-4 .initial:n  = { nosep, label=\Alph*., first=\small, font=\small },
4231      print-* .code:n     = \keys_precompile:neN { enumext / enumext* }
4232                              { \__enumext_filter_save_key:n {#1} }
4233                              \l__enumext_print_keyans_vii_tl, % starred nested
4234      print-* .initial:n  = { nosep, label=\arabic*., first=\small, font=\small },
4235      }
```

💣 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, \item* and \anspic*. Within a group we will run our *"precompiled keys"* and then call the internal function __enumext_-printkeyans:nnn.

```
4236 \NewDocumentCommand \printkeyans { s O{} m }
4237   {
4238     \group_begin:
4239       \tl_use:N \l__enumext_print_keyans_i_tl
4240       \tl_use:N \l__enumext_print_keyans_ii_tl
4241       \tl_use:N \l__enumext_print_keyans_iii_tl
4242       \tl_use:N \l__enumext_print_keyans_iv_tl
4243       \tl_use:N \l__enumext_print_keyans_vii_tl
4244       \__enumext_printkeyans:nnn { #1 } { #2 } { #3 }
4245     \group_end:
4246   }
```

(*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.

```
4247 \cs_new_protected:Npn \__enumext_printkeyans:nnn #1 #2 #3
4248   {
4249     \seq_if_exist:cTF { g__enumext_#3_seq }
4250       {
4251         \seq_if_empty:cF { g__enumext_#3_seq }
4252           {
4253             %%\seq_show:c { g__enumext_#3_seq }
```

If the starred if it 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 and then will map the ⟨sequence⟩

```
4254             \bool_if:nTF {#1}
4255               {
4256                 \seq_if_in:cnTF { g__enumext_#3_seq } { \end{enumext*} }
4257                   {
4258                     \msg_error:nnnn { enumext } { print-starred } {#3} { enumext* }
4259                   }
4260                   {
4261                     \tl_use:N \l__enumext_print_keyans_starred_tl
4262                     \begin{enumext*}[#2]
4263                       \seq_map_inline:cn { g__enumext_#3_seq } { ##1 }
4264                     \end{enumext*}
4265                   }
4266               }
```

Otherwise it will open the environment enumext passing the optional argument to the first level and then map the ⟨sequence⟩.

```
4267               {
4268                 \begin{enumext}[#2]
4269                   \seq_map_inline:cn { g__enumext_#3_seq } { ##1 }
4270                 \end{enumext}
4271               }
4272           }
4273       }
```

```
4274          {
4275            \msg_error:nnn { enumext } { undefined-storage-anskey } {#3}
4276          }
4277      }
```

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

11.41   The command \setenumext

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

```
4278  \keys_define:nn { enumext / meta-families }
4279      {
4280        enumext-1 .code:n = { \keys_set:nn { enumext / level-1  } {#1} } ,
4281        enumext-2 .code:n = { \keys_set:nn { enumext / level-2  } {#1} } ,
4282        enumext-3 .code:n = { \keys_set:nn { enumext / level-3  } {#1} } ,
4283        enumext-4 .code:n = { \keys_set:nn { enumext / level-4  } {#1} } ,
4284        keyans    .code:n = { \keys_set:nn { enumext / keyans   } {#1} } ,
4285        enumext*  .code:n = { \keys_set:nn { enumext / enumext* } {#1} } ,
4286        keyans*   .code:n = { \keys_set:nn { enumext / keyans*  } {#1} } ,
4287        print*    .code:n = { \keys_set:nn { enumext / print    } { print*  = {#1} } } ,
4288        print-1   .code:n = { \keys_set:nn { enumext / print    } { print-1 = {#1} } } ,
4289        print-2   .code:n = { \keys_set:nn { enumext / print    } { print-2 = {#1} } } ,
4290        print-3   .code:n = { \keys_set:nn { enumext / print    } { print-3 = {#1} } } ,
4291        print-4   .code:n = { \keys_set:nn { enumext / print    } { print-4 = {#1} } } ,
4292        print-*   .code:n = { \keys_set:nn { enumext / print    } { print-* = {#1} } } ,
4293        unknown   .code:n = { \msg_error:nn { enumext } { unknown-key-family } } ,
4294      }
```

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

```
4295  \seq_const_from_clist:Nn \c__enumext_all_families_seq
4296      {
4297        enumext-1, enumext-2, enumext-3, enumext-4, keyans, enumext*,
4298        keyans*, print-1, print-2, print-3, print-4, print-*, print*,
4299      }
```

\setenumext   Now we define the user command \setenumext.

```
4300  \NewDocumentCommand \setenumext { O{enumext,1} +m }
4301      {
4302        \tl_if_novalue:nTF {#1}
4303          {
4304            \seq_map_inline:Nn \c__enumext_all_families_seq
4305          }
4306          {
4307            \seq_clear:N \l__enumext_setkey_tmpa_seq
4308            \seq_set_from_clist:Nn \l__enumext_setkey_tmpb_seq {#1}
4309            \int_set:Nn \l__enumext_setkey_tmpa_int
4310              {
4311                \seq_count:N \l__enumext_setkey_tmpb_seq
4312              }
4313            \int_compare:nNnTF { \l__enumext_setkey_tmpa_int } > { 1 }
4314              {
4315                \seq_pop_left:NN \l__enumext_setkey_tmpb_seq \l__enumext_setkey_tmpa_tl
4316                \seq_map_function:NN \l__enumext_setkey_tmpb_seq \__enumext_set_parse:n
4317                \seq_set_map_e:NNn \l__enumext_setkey_tmpa_seq \l__enumext_setkey_tmpa_seq
4318                  {
4319                    \tl_use:N \l__enumext_setkey_tmpa_tl - ##1
4320                  }
4321              }
4322              {
4323                \seq_put_right:Ne \l__enumext_setkey_tmpa_seq { \tl_trim_spaces:n {#1} }
4324              }
4325            \seq_if_empty:NTF \l__enumext_setkey_tmpa_seq
4326              { \seq_map_inline:Nn \c__enumext_all_families_seq }
4327              { \seq_map_inline:Nn \l__enumext_setkey_tmpa_seq }
4328          }
4329          {
4330            \keys_set:nn { enumext / meta-families } { ##1 = {#2} }
4331          }
4332      }
```

(*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.

```
4333 \cs_new_protected:Npn \__enumext_set_parse:n #1
4334   {
4335     \tl_set:Ne \l__enumext_setkey_tmpb_tl { \tl_trim_spaces:n {#1} }
4336     \clist_map_inline:nn { 0, 1, 2, 3, 4, * } % <- max level
4337       { \tl_remove_all:Nn \l__enumext_setkey_tmpb_tl {##1} }
4338     \tl_if_empty:NTF \l__enumext_setkey_tmpb_tl
4339       {
4340         \seq_put_right:Ne \l__enumext_setkey_tmpa_seq
4341           { \tl_trim_spaces:n {#1} }
4342       }
4343       { \__enumext_set_error:nn {#1} { } }
4344   }
4345 \cs_new_protected:Npn \__enumext_set_error:nn #1 #2
4346   { \msg_error:nnn { enumext } { invalid-key } {#1} {#2} }
```

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

11.42   Messages

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

```
4347 \msg_new:nnn { enumext } { package-load }
4348   {
4349     The ~ '#1' ~ package ~ is ~ already ~ loaded.
4350   }
4351 \msg_new:nnn { enumext } { package-not-load }
4352   {
4353     The ~ '#1' ~ package ~ will ~ be ~ loaded ~ as ~ a ~ dependency.
4354   }
4355 \msg_new:nnn { enumext } { package-load-foot }
4356   {
4357     The ~ '#1' ~ package ~ is ~ loaded ~ with ~ the ~ option ~ '#2'.
4358   }
```

Message used in the creation of counters by enumext package.

```
4359 \msg_new:nnn { enumext } { counters }
4360   {
4361     The ~ counter ~ '#1' ~ is ~ already ~ defined ~ by ~ some ~ \\
4362     package ~ or ~ macro, ~ it ~ cannot ~ be ~ continued.
4363   }
```

Message used by reserved anskeyenv environment by enumext package.

```
4364 \msg_new:nnn { enumext } { anskeyenv-defined }
4365   {
4366     The ~ '#1' ~ environment ~ is ~ reserved ~ by ~\\
4367     'enumext'~ package, ~ it ~ cannot ~ be ~ continued.
4368   }
```

Message used in the creation of ⟨*prop list*⟩ by enumext package.

```
4369 \msg_new:nnn { enumext } { store-prop }
4370   {
4371     * ~ Package ~ enumext: ~ Creating ~ \c_backslash_str g__enumext_#1_prop ~ \msg_line_context:.
4372   }
4373 \msg_new:nnn { enumext } { store-seq }
4374   {
4375     * ~ Package ~ enumext: ~ Creating ~ \c_backslash_str g__enumext_#1_seq ~ \msg_line_context:.
4376   }
4377 \msg_new:nnn { enumext } { store-int }
4378   {
4379     * ~ Package ~ enumext: ~ Creating ~ \c_backslash_str g__enumext_resume_#1_int ~ \msg_line_con
4380   }
4381 \msg_new:nnn { enumext } { prop-seq-int-hook }
4382   {
4383     * ~ Package ~ enumext: ~ Elements ~ in ~ \c_backslash_str g__enumext_#1_prop ~ = ~ #2.\\
4384     * ~ Package ~ enumext: ~ Elements ~ in ~ \c_backslash_str g__enumext_#1_seq ~ = ~ #3.\\
4385     * ~ Package ~ enumext: ~ Value ~ off ~ \c_backslash_str g__enumext_resume_#1_int ~ = ~ #4.
4386   }
4387 \msg_new:nnn { enumext } { item-answer-hook }
4388   {
4389     * ~ Package ~ enumext: ~ Value ~ off ~ \c_backslash_str g__enumext_item_number_int ~ = ~ #1.\\
4390     * ~ Package ~ enumext: ~ Value ~ off ~ \c_backslash_str g__enumext_item_anskey_int ~ = ~ #2.\\
4391     * ~ Package ~ enumext: ~ Difference ~ item_number_int ~ - ~ item_anskey_int ~ = ~ #3.
4392   }
```

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

```
4393 \msg_new:nnn { enumext } { invalid-key }
4394   {
4395     The ~ key ~ '#1' ~ is ~ not ~ know ~ the ~ level ~ #2.
4396   }
4397 \msg_new:nnn { enumext } { unknown-key-family }
4398   {
4399     Unknown~key~family~`\l_keys_key_str'~for~enumext.
4400   }
```

Messages used in length calculation.

```
4401 \msg_new:nnn { enumext } { width-negative }
4402   {
4403     Ignoring ~ negative ~ value ~ '#1=#2' ~ \msg_line_context:.\\
4404     The ~ key ~ '#1'~ accepts ~ values  ~ >= ~ 0pt.
4405   }
4406 \msg_new:nnn { enumext } { width-zero }
4407   {
4408     Invalid ~ '#1=#2' ~ \msg_line_context:.\\
4409     The ~ key ~ '#1'~ accepts ~ values  ~ > ~ 0pt.
4410   }
```

Messages used by `show-length` key in `enumext`.

```
4411 \msg_new:nnn { enumext } { list-lengths }
4412   {
4413     **** ~ Lengths ~ used ~ by ~ 'enumext' ~ level ~ '#2' ~ \msg_line_context:~\c_space_tl ****\\
4414     \__enumext_show_length:nnn { dim  } { labelsep     } {#1}
4415     \__enumext_show_length:nnn { dim  } { labelwidth   } {#1}
4416     \__enumext_show_length:nnn { dim  } { itemindent   } {#1}
4417     \__enumext_show_length:nnn { dim  } { leftmargin   } {#1}
4418     \__enumext_show_length:nnn { dim  } { rightmargin  } {#1}
4419     \__enumext_show_length:nnn { dim  } { listparindent } {#1}
4420     \__enumext_show_length:nnn { skip } { topsep     } {#1}
4421     \__enumext_show_length:nnn { skip } { parsep     } {#1}
4422     \__enumext_show_length:nnn { skip } { partopsep } {#1}
4423     \__enumext_show_length:nnn { skip } { itemsep    } {#1}
4424     **************************************************
4425   }
```

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

```
4426 \msg_new:nnn { enumext } { list-lengths-not-nested }
4427   {
4428     **** ~ Lengths ~ used ~ by ~ '#2' ~ environment ~ \msg_line_context:~\c_space_tl ****\\
4429     \__enumext_show_length:nnn { dim  } { labelsep     } {#1}
4430     \__enumext_show_length:nnn { dim  } { labelwidth   } {#1}
4431     \__enumext_show_length:nnn { dim  } { itemindent   } {#1}
4432     \__enumext_show_length:nnn { dim  } { leftmargin   } {#1}
4433     \__enumext_show_length:nnn { dim  } { rightmargin  } {#1}
4434     \__enumext_show_length:nnn { dim  } { listparindent } {#1}
4435     \__enumext_show_length:nnn { skip } { topsep     } {#1}
4436     \__enumext_show_length:nnn { skip } { parsep     } {#1}
4437     \__enumext_show_length:nnn { skip } { partopsep } {#1}
4438     \__enumext_show_length:nnn { skip } { itemsep    } {#1}
4439     **************************************************
4440   }
```

Messages used by `ref` key.

```
4441 \msg_new:nnn { enumext } { key-ref-empty }
4442   {
4443     Key ~ 'ref' ~ need ~ a ~ value ~ in ~ '#1'~ \msg_line_context:.
4444   }
```

Messages used by `save-ans` key.

```
4445 \msg_new:nnn { enumext } { save-ans-empty }
4446   {
4447     Key ~ 'save-ans' ~ need ~ a ~ value ~ in ~ '#1'~ \msg_line_context:.
4448   }
4449 \msg_new:nnn { enumext } { save-ans-log }
4450   {
4451     * ~ Package ~ enumext: ~ Start ~ \c_left_brace_str#1\c_right_brace_str \c_space_tl with ~ save-
   ans=#2 ~ \msg_line_context:.
4452   }
4453 \msg_new:nnn { enumext } { save-ans-log-hook }
```

```
4454       {
4455         * ~ Package ~ enumext: ~ Stop ~ \c_left_brace_str#1\c_right_brace_str \c_space_tl with ~ save-
             ans=#2 ~ \msg_line_context:.
4456       }
4457 \msg_new:nnn { enumext } { save-ans-hook }
4458       {
4459         Stop ~ storing ~ for ~ 'save-ans=#1' ~ \msg_line_context:.
4460       }
```

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

```
4461 \msg_new:nnn { enumext } { need-save-ans }
4462       {
4463         Key ~ '#1'~ works ~ only ~ with ~ the ~ 'save-ans' ~ key ~ in ~ '#2'~ \msg_line_context:.
4464       }
4465 \msg_new:nnn { enumext } { items-same-answer }
4466       {
4467         ****************************************\\
4468         * ~ Package ~ enumext: ~ Checking ~ answers ~ in ~ '#1' ~ for ~ \c_left_brace_str #2 \c_right_
4469         * ~ started ~ #3 ~ and ~ close ~ \msg_line_context: : ~ 'OK', ~ all ~ items ~ with ~ answer.\'
4470         ****************************************
4471       }
4472 \msg_new:nnn { enumext } { item-greater-answer }
4473       {
4474         Checking ~ answers ~ in ~ '#1' ~ for ~ \c_left_brace_str #2 \c_right_brace_str\\
4475         started ~ #3 ~ and ~ close ~ \msg_line_context: : ~'NOT ~ OK'\\
4476         Items ~ > ~ Answers.
4477       }
4478 \msg_new:nnn { enumext } { item-less-answer }
4479       {
4480         Checking ~ answers ~ in ~ '#1' ~ for ~ \c_left_brace_str #2 \c_right_brace_str\\
4481         started ~ #3 ~ and ~ close ~ \msg_line_context: : ~'NOT ~ OK'\\
4482         Items ~ < ~ Answers.
4483       }
```

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

```
4484 \msg_new:nnn { enumext } { missing-starred }
4485       {
4486         Missing ~ '\c_backslash_str #1*' ~ #2.
4487       }
4488 \msg_new:nnn { enumext } { many-starred }
4489       {
4490         Many ~ '\c_backslash_str #1*' ~ #2.
4491       }
```

Messages used by \printkeyans* command.

```
4492 \msg_new:nnn { enumext } { print-starred }
4493       {
4494         \c_backslash_str printkeyans*:~ The ~ sequence ~ '#1' ~ already ~ contains ~
4495         #2 ~ environment ~  \msg_line_context:.
4496       }
```

Message for the nesting depth of the environment enumext.

```
4497 \msg_new:nnn { enumext } { list-too-deep }
4498       {
4499         Too ~ deep ~ nesting  ~ for ~ 'enumext' ~ \msg_line_context:.~ \\
4500         The ~ maximum  ~ level  ~ of  ~ nesting  ~ is ~ 4.
4501       }
```

Messages used by \anskey and \anspic commands.

```
4502 \msg_new:nnn { enumext } { anskey-empty-arg }
4503       {
4504         Can't ~ store ~ empty ~ content ~ ~ \msg_line_context:.
4505       }
4506 \msg_new:nnn { enumext } { anskey-wrong-place }
4507       {
4508         Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:.~ \\
4509         '\c_backslash_str #1' ~ works ~ in ~ the ~ environment ~ '#2'.
4510       }
4511 \msg_new:nnn { enumext } { anskey-nested }
4512       {
4513         The ~ command ~ \c_backslash_str anskey~ can't ~ be ~ nested ~ \msg_line_context:.
4514       }
4515 \msg_new:nnn { enumext } { anskey-nested-env }
```

```
4516      {
4517        The ~ environment ~ anskey* ~ can't ~ be ~ nested ~ \msg_line_context:.
4518      }
4519 \msg_new:nnn { enumext } { anspic-wrong-place }
4520      {
4521        Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:.~ \\
4522        '\c_backslash_str #1' ~ works ~ in ~ the ~ environment ~ '#2'.
4523      }
4524 \msg_new:nnn { enumext } { command-wrong-place }
4525      {
4526        Wrong ~ place ~ for ~ command ~ \c_backslash_str #1' ~ \msg_line_context:.~ \\
4527        '\c_backslash_str #1' ~ works ~ outside ~ the ~ environment ~ '#2'.
4528      }
```

Messages used by keyans and keyanspic environment.

```
4529 \msg_new:nnn { enumext } { keyans-nested }
4530      {
4531        The ~ environment ~ 'keyans' ~ can't ~ be  ~ nested  ~ \msg_line_context:.
4532      }
4533 \msg_new:nnn { enumext } { keyans-wrong-level }
4534      {
4535        Wrong ~ level ~ position ~ for ~ 'keyans' ~ \msg_line_context:.~ \\
4536        The ~ environment ~ 'keyans' ~ can ~ only ~ be ~ in ~ the ~ first ~ level.
4537      }
4538 \msg_new:nnn { enumext } { wrong-place }
4539      {
4540        Wrong ~ place ~ for ~ '#1' ~ environment ~\msg_line_context:.~ \\
4541        '#1' ~ is ~ only ~ found ~ with ~ '#2' ~  in  ~  'enumext.
4542      }
4543 \msg_new:nnn { enumext } { keyanspic-nested }
4544      {
4545        The ~ environment ~ 'keyanspic' ~ can't ~ be  ~ nested~ \msg_line_context:.~.
4546      }
4547 \msg_new:nnn { enumext } { keyanspic-wrong-level }
4548      {
4549        Wrong ~ level ~ position ~ for ~ 'keyanspic' ~ \msg_line_context:.~ \\
4550        The ~ environment ~ 'keyans' ~ can ~ only ~ be ~ in ~ the ~ first ~ level.
4551      }
```

Messages used by \getkeyans command.

```
4552 \msg_new:nnn { enumext } { undefined-storage-anskey }
4553      {
4554        Storage ~ named ~ '#1' ~ is ~ not ~ defined ~ \msg_line_context:.
4555      }
```

Messages used by \miniright command.

```
4556 \msg_new:nnn { enumext } { missing-miniright }
4557      {
4558        Missing ~ '\c_backslash_str miniright' ~ in ~ \msg_line_context:.\\
4559        The ~ key ~ 'mini-env' ~ need ~ '\c_backslash_str miniright'.
4560      }
4561 \msg_new:nnn { enumext } { wrong-miniright-place }
4562      {
4563        Wrong ~ place ~ for ~ '\c_backslash_str miniright' ~ \msg_line_context:.~ \\
4564        Works ~ in ~ 'enumext' ~ and ~ 'keyans' ~ with ~ key ~ 'mini-env'.
4565      }
4566 \msg_new:nnn { enumext } { wrong-miniright-use }
4567      {
4568        Wrong ~ use ~ for ~ '\c_backslash_str miniright' ~ \msg_line_context:.~ \\
4569        '\c_backslash_str miniright' ~ need ~ a ~ key ~ 'mini-env'.
4570      }
```

Messages used by enumext* and keyans* environments.

```
4571 \msg_new:nnn { enumext } { nested }
4572      {
4573        The ~ starred ~ environment ~ can't ~ be ~ nested ~ \msg_line_context:.
4574      }
4575 \msg_new:nnn { enumext } { item-joined }
4576      {
4577        Items ~ joined ~ (#1) ~ > ~ #2  ~ columns ~\msg_line_context:.
4578      }
4579 \msg_new:nnn { enumext } { item-joined-columns }
4580      {
```

```
4581      Not ~ space ~ to ~ join ~ items ~ (#1) ~ > ~ #2 ~\msg_line_context:.
4582    }
```

11.43 Finish package

Finish package implementation.

```
4583 \file_input_stop:
4584 ⟨/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.