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

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

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

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

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

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

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

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

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

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

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

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

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

4. Question with image and label below:

A)   B)   C)

D)   E)

5. Question with image on left side:

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

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

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

4. 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*`, `anskey*`, `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.
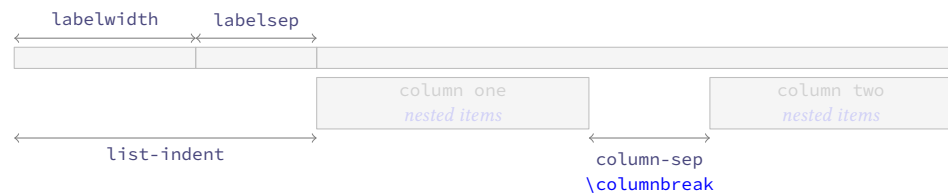

Figure 4: Representation of the two column output for a nested level in `enumext` environment.

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 ninth
  \item The tenth
\end{enumext*}
```

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

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

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

8. The tenth

3 The command \setenumext

\setenumext

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

\setenumext[⟨keyans*⟩]{⟨key = val⟩}
\setenumext[⟨print, level⟩]{⟨key = val⟩}
\setenumext[⟨print, *⟩]{⟨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 `anskey*`, `keyans`, `keyans*` and `keyanspic` are available.

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

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

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

5.1 Keys for storage system

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

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

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

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

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

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

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

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

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

5.1.1 Keys for `label` and `ref`

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

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

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

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

5.1.2 Keys for `wrap` and `display`

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

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

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

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

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

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

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

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

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

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

5.1.3 Keys for debug and checking

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

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

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

Enables the *checking answer* mechanism displaying an appropriate message on the terminal. This key works under the logic that each `\item` or `\item*` that does not open an inner level or nested environment contains *"only one answer"* or *"only one execution"* of the `\anskey` or `anskey*`. It is intended to be used in conjunction with the `no-store` key.

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

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

5.2 The command `\anskey`

`\anskey` | `\anskey[`⟨*keys*⟩`]{`⟨*content*⟩`}`

The command `\anskey` takes a mandatory 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 anskey*

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

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

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

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

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

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

Example

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

★ 5. Text containing our instructions or questions.
  [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*

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

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

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

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

5.4.1 The \item* in keyans and keyans*

`\item*`

\item*
\item*[⟨content⟩]

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

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

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

Example

```
\begin{enumext}[save-ans=test,columns=2,show-ans=true]
  \item Text containing a question.
    \begin{keyans*}[nosep,columns=2]
      \item Choice
      \item* Correct choice
      \item Choice
      \item Choice
      \item Choice
    \end{keyans*}
  \item Text containing a question and image.
    \begin{keyans}[nosep,mini-env={0.4\linewidth}]
      \item Choice
      \item Choice
      \item Choice
      \item Choice
      \item*[⟨note⟩] Correct choice
      \miniright
      \includegraphics[scale=0.25]{example-image-a}
      Some text
    \end{keyans}
\end{enumext}
```

1. Text containing a question.
   - A) Choice
   - * B) Correct choice
   - C) Choice
   - D) Choice
   - E) Choice

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

Some text
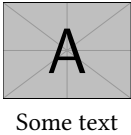
5.5 The environment keyanspic

`keyanspic`

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

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

A)   B)   C)

D)   * E)[note]
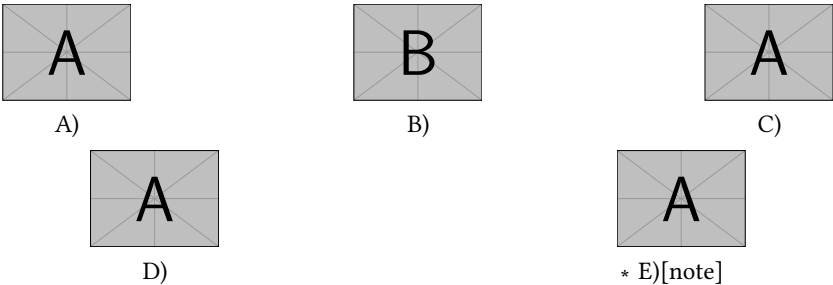
5.6 Printing stored content

5.6.1 The command \getkeyans

\getkeyans

\getkeyans{⟨*store name : position*⟩}

The command \getkeyans prints the *"stored content"* in *prop list* {⟨*store name*⟩} defined by `save-ans` key in the ⟨*position*⟩ returned by the `show-pos` key. The *"stored content"* can only be accessed *after* it is stored, if {⟨*store name*⟩} does not exist the command will return an error.

The form taken by the argument {⟨*store name : position*⟩} is the same as that used to generate the *"internal label and ref"* system when `save-ref` key are active, so to refer to a *"stored content"*. For example \getkeyans{test:4} will return the *"stored content"* at position 4 of the environment in which the key `save-ans=test` was set.

5.6.2 The command \printkeyans

\printkeyans

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

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

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

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

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

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

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

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

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

Example

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

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

  \item Related to Linux

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

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

\printkeyans{sample}
```

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

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

2. True False

   (a) LaTeX2e is cool?

    [2] Very True!

3. Related to Linux

   (a) You use linux?

[3] Yes

   (b) Rate the following package and class

    i. xsim

    [4] very good

    ii. exsheets

    [5] obsolete

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

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

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

Ⓐ

Ⓑ

Ⓒ

Ⓓ

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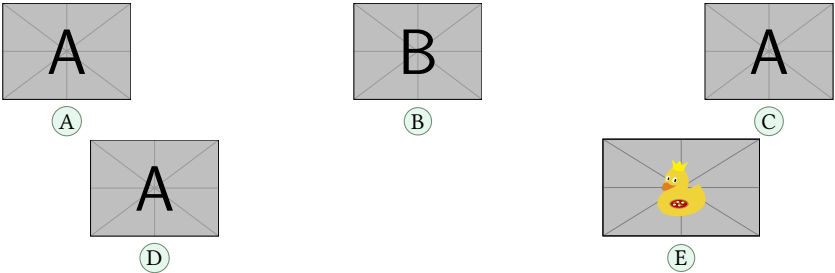
5. Question with image on left side:
   - Ⓐ value
   - Ⓑ value
   - Ⓒ value
   - Ⓓ correct
   - Ⓔ value

Test keys

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

Example 4

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

🦆 Factor $x^2 - 2x + 1$

🦆 Factor $3x + 3y + 3z$

The following questions need to be cuaqtified :)

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

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

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

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

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

Example 5

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

1

Which choice best describes what happens in the passage?

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

3

Which choice best describes what happens in the passage?

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

2

Which choice best describes what happens in the passage?

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

4

Which choice best describes what happens in the passage?

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

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

7 The way of non-enumerated lists

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

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

Fake `itemize` environment

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

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

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

Fake `description` environment

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

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

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

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

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

Description indented by label

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

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

and then use `labelsep=4`pt`,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 `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 - 4`pt`}`.

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 - 4`pt`}` 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$_\varepsilon$: 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.*)

`\l__enumext_current_widest_dim`
`\g__enumext_counter_styles_tl`
`\g__enumext_widest_label_tl`
`\l__enumext_label_width_by_box`

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

```
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.*)

`\l__enumext_leftmargin_tmp_X_bool`
`\l__enumext_leftmargin_tmp_X_dim`
`\l__enumext_leftmargin_X_dim`
`\l__enumext_itemindent_X_dim`

The boolean variable `\l__enumext_leftmargin_tmp_X_bool` and the dimensional variable `\l__enumext_leftmargin_tmp_X_dim` are used by the list-indent key (§11.15). The variables `\l__enumext_leftmargin_X_dim` and `\l__enumext_itemindent_X_dim` are used and set by the function `\__enumext_calc_hspace:NNNNNNNNNNNN` (§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.*)

`\l__enumext_multicols_above_X_skip`
`\l__enumext_multicols_below_X_skip`

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`.)

`\g__enumext_minipage_stat_int`
`\l__enumext_minipage_left_skip`
`\l__enumext_minipage_right_skip`
`\l__enumext_minipage_after_skip`
`\g__enumext_minipage_right_skip`
`\g__enumext_minipage_after_skip`
`\l__enumext_minipage_left_X_dim`
`\l__enumext_minipage_active_X_bool`

Internal variables used by `\miniright` command (§11.20.4) 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.*)

`\l__enumext_wrap_label_X_bool`
`\l__enumext_wrap_label_opt_X_bool`
`\l__enumext_start_X_int`
`\l__enumext_fake_item_indent_X_tl`
`\l__enumext_label_fill_left_X_tl`
`\l__enumext_label_fill_right_X_tl`
`\l__enumext_vspace_a_star_X_bool`
`\l__enumext_vspace_b_star_X_bool`

The integer variable `\l__enumext_start_X_int` are used by the start key (§11.13), the token list `\l__enumext_fake_item_indent_X_tl` is used by itemindent key (§11.15.1), 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.)

The variable \l__enumext_store_active_bool setting by save-ans key (§11.23.1) activates all the mechanism related to \anskey, anskey*, keyans, keyans* and keyanspic environments.

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

The variable \l__enumext_store_anskey_arg_tl save the *argument* of \anskey (§11.26) and the variables \l__enumext_store_anskey_env_tl, \l__enumext_store_anskey_opt_tl save the ⟨*body*⟩ and ⟨*keys*⟩ of the environment anskey* (§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.)

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.)

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.)

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.)

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 a 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 / unknown .code:n =
508                            \msg_error:nneee { enumext } { unknown-choice }
509                              { align } { left, ~ right, ~  center } { \exp_not:n {##1} },
510          align .initial:n  = left,
511          align .value_required:n  = true,
512        }
513    }
514  \clist_map_inline:nn
515    {
516      {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {keyans}{v}
517    }
518    { \__enumext_tmp:nn #1 }

519  \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
520    {
521      \keys_define:nn { enumext / #1 }
522        {
523          align .choice:,
524          align / left     .code:n = \str_set:cn { l__enumext_align_label_#2_str } { l },
525          align / right    .code:n = \str_set:cn { l__enumext_align_label_#2_str } { r },
526          align / center   .code:n = \str_set:cn { l__enumext_align_label_#2_str } { c },
527          align / unknown .code:n =
528                            \msg_error:nneee { enumext } { unknown-choice }
529                              { align } { left, ~ right, ~  center } { \exp_not:n {##1} },
530          align .initial:n = left,
531          align .value_required:n = true,
532        }
533    }
534  \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 Here we set the default ⟨*labels*⟩ of the *four levels* of enumext environment, along with the default value for
ref `labelwidth` key and `ref` key.

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

```
535  \cs_set_protected:Npn \__enumext_tmp:nnn #1 #2 #3
536    {
537      \keys_define:nn { enumext / #1 }
538        {
539          label .code:n    = {
540                               \__enumext_label_style:cvn { l__enumext_label_#2_tl }
541                                 { l__enumext_counter_#2_tl } {##1}
542                               \dim_set_eq:cN  { l__enumext_labelwidth_#2_dim }
543                                 \l__enumext_current_widest_dim
```

```
544                              },
545              label  .initial:n = #3,
546              label  .value_required:n = true,
547              ref    .code:n    = \__enumext_standar_ref:n {##1},
548              ref    .value_required:n = true,
549            }
550      }
551  \__enumext_tmp:nnn { level-1 } {   i } { \arabic*.}
552  \__enumext_tmp:nnn { level-2 } {  ii } { (\alph*) }
553  \__enumext_tmp:nnn { level-3 } { iii } { \roman*. }
554  \__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.

```
555  \cs_new_protected:Npn \__enumext_standar_ref:n #1
556    {
557      \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
558      \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
559        {
560          \msg_error:nnn { enumext } { key-ref-empty } { enumext }
561        }
562        {
563          \tl_set_eq:Nc
564            \l__enumext_ref_the_count_tl { l__enumext_counter_ \__enumext_level: _tl }
565          \__enumext_regex_counter_style:
566          \tl_set_eq:Nc
567            \l__enumext_ref_the_count_tl { l__enumext_the_counter_ \__enumext_level: _tl }
568          \tl_put_right:ce { l__enumext_renew_the_count_ \__enumext_level: _tl }
569            {
570              \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl }
571                { \exp_not:V \l__enumext_ref_key_arg_tl }
572            }
573        }
574    }
```

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

```
575  \cs_new_protected:Nn \__enumext_standar_ref:
576    {
577      \tl_if_empty:cF { l__enumext_renew_the_count_ \__enumext_level: _tl }
578        {
579          \tl_use:c { l__enumext_renew_the_count_ \__enumext_level: _tl }
580        }
581    }
```

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

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

label
ref

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

\l__enumext_label_vii_tl
\l__enumext_label_viii_tl

```
582  \cs_set_protected:Npn \__enumext_tmp:nnn #1 #2 #3
583    {
584      \keys_define:nn { enumext / #1 }
585        {
586          label .code:n    = {
587                                \__enumext_label_style:cvn { l__enumext_label_#2_tl }
588                                  { l__enumext_counter_#2_tl } {##1}
589                                \dim_set_eq:cN  { l__enumext_labelwidth_#2_dim }
590                                  \l__enumext_current_widest_dim
591                                },
592          label .initial:n = #3,
593          label .value_required:n = true,
594          ref   .code:n    = \__enumext_starred_ref:n {##1},
595          ref   .value_required:n = true,
596        }
```

```
597      }
598  \__enumext_tmp:nnn { enumext* } {  vii } { \arabic*.}
599  \__enumext_tmp:nnn { keyans*  } { viii } { \Alph*) }
```

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

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

```
600  \cs_new_protected:Npn \__enumext_starred_ref:n #1
601    {
602      \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
603      \int_compare:nNnT { \l__enumext_level_h_int } = { 1 }
604        {
605          \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
606            {
607              \msg_error:nnn { enumext } { key-ref-empty } { enumext* }
608            }
609            {
610              \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_counter_vii_tl
611              \__enumext_regex_counter_style:
612              \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_the_counter_vii_tl
613              \tl_put_right:Ne \l__enumext_renew_the_count_vii_tl
614                {
615                  \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl }
616                    { \exp_not:V \l__enumext_ref_key_arg_tl }
617                }
618            }
619        }
620      \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
621        {
622          \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
623            {
624              \msg_error:nnn { enumext } { key-ref-empty } { keyans* }
625            }
626            {
627              \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_counter_viii_tl
628              \__enumext_regex_counter_style:
629              \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_the_counter_viii_tl
630              \tl_put_right:Ne \l__enumext_renew_the_count_viii_tl
631                {
632                  \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl }
633                    { \exp_not:V \l__enumext_ref_key_arg_tl }
634                }
635            }
636        }
637    }
```

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.

```
638  \cs_new_protected:Nn \__enumext_starred_ref:
639    {
640      \int_compare:nNnT { \l__enumext_level_h_int } = { 1 }
641        {
642          \tl_if_empty:NF \l__enumext_renew_the_count_vii_tl
643            {
644              \tl_use:N \l__enumext_renew_the_count_vii_tl
645            }
646        }
647      \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
648        {
649          \tl_if_empty:NF \l__enumext_renew_the_count_viii_tl
650            {
651              \tl_use:N \l__enumext_renew_the_count_viii_tl
652            }
653        }
654    }
```

(*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.

```
655 \keys_define:nn { enumext / keyans }
656   {
657     label .code:n    = {
658                          \__enumext_label_style:cvn { l__enumext_label_v_tl }
659                            { l__enumext_counter_v_tl } {#1}
660                          \dim_set_eq:cN  { l__enumext_labelwidth_v_dim }
661                            \l__enumext_current_widest_dim
662                          \__enumext_label_style:cvn { l__enumext_label_vi_tl }
663                            { l__enumext_counter_vi_tl } {#1}
664                          \dim_set_eq:cN  { l__enumext_labelwidth_v_dim }
665                            \l__enumext_current_widest_dim
666                        },
667     label .initial:n = \Alph*),
668     label .value_required:n = true,
669     ref   .code:n    = \__enumext_keyans_ref:n {#1},
670     ref   .value_required:n = true,
671   }
```

(*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`.

```
672 \cs_new_protected:Npn \__enumext_keyans_ref:n #1
673   {
674     \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
675     \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
676       {
677         \msg_error:nnn { enumext } { key-ref-empty } { keyans }
678       }
679       {
680         \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_counter_v_tl
681         \__enumext_regex_counter_style:
682         \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_the_counter_v_tl
683         \tl_put_right:Ne \l__enumext_renew_the_count_v_tl
684           {
685             \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl }
686               { \exp_not:V \l__enumext_ref_key_arg_tl }
687           }
688       }
689   }
```

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

```
690 \cs_new_protected:Nn \__enumext_keyans_ref:
691   {
692     \tl_if_empty:NF \l__enumext_renew_the_count_v_tl
693       {
694         \tl_use:N \l__enumext_renew_the_count_v_tl
695       }
696   }
```

(*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.

```
697 \cs_new_protected:Npn \__enumext_start_from:NNn #1 #2 #3
698   {
699     \__enumext_if_is_int:nTF { #3 }
700       {
701         \int_set:Nn #2 {#3}
702       }
```

```
703        {
704          \regex_match:nVT { \c{Alph} | \c{alph} } {#1}
705            { \int_set:Nn #2 { \int_from_alph:n {#3} } }
706          \regex_match:nVT { \c{Roman} | \c{roman} } {#1}
707            { \int_set:Nn #2  { \int_from_roman:n {#3} } }
708        }
709    }
710  \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.

```
711  \cs_new_protected:Npn \__enumext_widest_from:nNNn #1 #2 #3 #4
712    {
713      \__enumext_if_is_int:nTF {#4}
714        {
715          \setcounter{enumX#1} { #4 }
716        }
717        {
718          \regex_match:nVT { \c{Alph} | \c{alph} } {#2}
719            { \setcounter{enumX#1} { \int_from_alph:n {#4} } }
720          \regex_match:nVT { \c{Roman} | \c{roman} } {#2}
721            { \setcounter{enumX#1} { \int_from_roman:n {#4} } }
722        }
723      \__enumext_label_width_by_box:cv
724        { l__enumext_labelwidth_#1_dim } { l__enumext_label_#1_tl }
725    }
726  \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.

```
727  \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
728    {
729      \keys_define:nn { enumext / #1 }
730        {
731          start  .code:n    = {
732                                \__enumext_start_from:ccn
733                                  { l__enumext_label_#2_tl }
734                                  { l__enumext_start_#2_int } {##1}
735                              },
736          start  .initial:n = 1,
737          widest .code:n    = {
738                                \__enumext_widest_from:nccn {#2}
739                                  { l__enumext_label_#2_tl }
740                                  { l__enumext_labelwidth_#2_dim } {##1}
741                              },
742          widest .value_required:n = true,
743          start  .value_required:n = true,
744        }
745    }
746  \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.

```
747  \cs_set_protected:Npn \__enumext_tmp:nnnnnn #1 #2 #3 #4 #5 #6
748    {
749      \keys_define:nn { enumext / #1 }
```

```
750       {
751         topsep     .skip_set:c = { l__enumext_topsep_#2_skip },
752         topsep     .initial:n  = {#3},
753         topsep     .value_required:n = true,
754         partopsep .skip_set:c = { l__enumext_partopsep_#2_skip },
755         partopsep .initial:n  = {#4},
756         partopsep .value_required:n = true,
757         parsep     .skip_set:c = { l__enumext_parsep_#2_skip },
758         parsep     .initial:n  = {#5},
759         parsep     .value_required:n = true,
760         itemsep    .skip_set:c = { l__enumext_itemsep_#2_skip },
761         itemsep    .initial:n  = {#6},
762         itemsep    .value_required:n = true,
763         noitemsep .meta:n      = { itemsep = 0pt, parsep = 0pt },
764         noitemsep .value_forbidden:n = true,
765         nosep      .meta:n      = {
766                                   itemsep = 0pt, parsep= 0pt,
767                                   topsep = 0pt, partopsep = 0pt,
768                                 },
769         nosep      .value_forbidden:n = true,
770       }
771   }
```

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

```
772 \__enumext_tmp:nnnnnn { level-1 } { i } { 8.0pt plus 2.0pt minus 4.0pt }
773   { 2.0pt plus 1.0pt minus 1.0pt } { 4.0pt plus 2.0pt minus 1.0pt }
774   { 4.0pt plus 2.0pt minus 1.0pt }
775 \__enumext_tmp:nnnnnn { level-2 } { ii } { 4.0pt plus 2.0pt minus 1.0pt }
776   { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
777   { 2.0pt plus 1.0pt minus 1.0pt }
778 \__enumext_tmp:nnnnnn { level-3 } { iii } { 2.0pt plus 1.0pt minus 1.0pt }
779   { 1.0pt minus 1.0pt }{ 0pt }{ 2.0pt plus 1.0pt minus 1.0pt }
780 \__enumext_tmp:nnnnnn { level-4 } { iv } { 2.0pt plus 1.0pt minus 1.0pt }
781   { 1.0pt minus 1.0pt }{ 0pt }{ 2.0pt plus 1.0pt minus 1.0pt }
782 \__enumext_tmp:nnnnnn { keyans  } { v }{ 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 }
785 \__enumext_tmp:nnnnnn { enumext* } { vii } { 8.0pt plus 2.0pt minus 4.0pt }
786   { 2.0pt plus 1.0pt minus 1.0pt } { 4.0pt plus 2.0pt minus 1.0pt }
787   { 4.0pt plus 2.0pt minus 1.0pt }
788 \__enumext_tmp:nnnnnn { keyans* } { viii } { 4.0pt plus 2.0pt minus 1.0pt }
789   { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
790   { 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.

```
791 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
792   {
793     \keys_define:nn { enumext / #1 }
794       {
795         itemindent    .dim_set:c = { l__enumext_fake_item_indent_#2_dim },
796         itemindent    .value_required:n = true,
797         rightmargin   .dim_set:c = { l__enumext_rightmargin_#2_dim },
798         rightmargin   .value_required:n = true,
799         listparindent .dim_set:c = { l__enumext_listparindent_#2_dim },
800         listparindent .value_required:n = true,
801         list-offset   .dim_set:c = { l__enumext_listoffset_#2_dim },
802         list-offset   .value_required:n = true,
803         list-indent   .code:n     =
804                         \bool_set_true:c { l__enumext_leftmargin_tmp_#2_bool }
805                         \dim_set:cn { l__enumext_leftmargin_tmp_#2_dim } {##1},
806         list-indent   .value_required:n = true,
807       }
808   }
809 \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.

```
810  \cs_set_protected:Npn \__enumext_tmp:n #1
811    {
812      \keys_define:nn { enumext / #1 } { list-indent .initial:n  = 0pt, }
813    }
814  \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.

```
815  \cs_set_protected:Nn \__enumext_fake_item:
816    {
817      \dim_compare:nNnT
818        { \dim_use:c { l__enumext_fake_item_indent_ \__enumext_level: _dim } }
819        >
820        { \c_zero_dim }
821        {
822          \tl_set:ce { l__enumext_fake_item_indent_ \__enumext_level: _tl }
823            {
824              \exp_not:N \mode_leave_vertical:
825              \exp_not:n { \skip_horizontal:n }
826                { \dim_use:c { l__enumext_fake_item_indent_ \__enumext_level: _dim } }
827              \ignorespaces
828            }
829        }
830    }
831  \cs_set_protected:Nn \__enumext_keyans_fake_item:
832    {
833      \dim_compare:nNnT
834        { \l__enumext_fake_item_indent_v_dim } > { \c_zero_dim }
835        {
836          \tl_set:Ne \l__enumext_fake_item_indent_v_tl
837            {
838              \exp_not:N \mode_leave_vertical:
839              \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_v_dim
840            }
841        }
842    }
843  \cs_set_protected:Nn \__enumext_fake_item_vii:
844    {
845      \dim_compare:nNnT
846        { \l__enumext_fake_item_indent_vii_dim } > { \c_zero_dim }
847        {
848          \tl_set:Ne \l__enumext_fake_item_indent_vii_tl
849            {
850              \exp_not:N \mode_leave_vertical:
851              \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_vii_dim
852            }
853        }
854    }
855  \cs_set_protected:Nn \__enumext_fake_item_viii:
856    {
857      \dim_compare:nNnT
858        { \l__enumext_fake_item_indent_viii_dim } > { \c_zero_dim }
859        {
860          \tl_set:Ne \l__enumext_fake_item_indent_viii_tl
861            {
862              \exp_not:N \mode_leave_vertical:
863              \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_viii_dim
864            }
865        }
866    }
```

(*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.

```
867 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
868   {
869     \keys_define:nn { enumext / #1 }
870       {
871         show-length .bool_set:c = { l__enumext_show_length_#2_bool },
872         show-length .initial:n  = false,
873       }
874   }
875 \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 Define and set before, before*, after and first keys for enumext, enumext*, keyans and keyans*
before* environments.
after
first
```
876 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
877   {
878     \keys_define:nn { enumext / #1 }
879       {
880         before  .tl_set:c   = { l__enumext_before_no_starred_key_#2_tl },
881         before  .value_required:n = true,
882         before* .tl_set:c   = { l__enumext_before_starred_key_#2_tl },
883         before* .value_required:n = true,
884         after   .tl_set:c   = { l__enumext_after_stop_list_#2_tl },
885         after   .value_required:n = true,
886         first   .tl_set:c   = { l__enumext_after_list_args_#2_tl },
887         first   .value_required:n = true,
888       }
889   }
890 \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: The function __enumext_before_args_exec: executes the {⟨*code*⟩} set by the before* key *"before"*
__enumext_before_keys_exec: the enumext environment is started. The {⟨*code*⟩} is executed *"without"* knowing any definition of the
__enumext_after_stop_list: *second argument* of the list.
__enumext_after_args_exec:
```
891 \cs_new_protected:Nn \__enumext_before_args_exec:
892   {
893     \tl_use:c { l__enumext_before_starred_key_ \__enumext_level: _tl }
894   }
```

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*⟩.

```
895 \cs_new_protected:Nn \__enumext_before_keys_exec:
896   {
897     \tl_use:c { l__enumext_before_no_starred_key_ \__enumext_level: _tl }
898   }
```

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

```
899 \cs_new_protected:Nn \__enumext_after_stop_list:
900   {
901     \tl_use:c { l__enumext_after_stop_list_ \__enumext_level: _tl }
902   }
```

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.

```
903 \cs_new_protected:Nn \__enumext_after_args_exec:
904   {
905     \tl_use:c { l__enumext_after_list_args_ \__enumext_level: _tl }
906   }
```

(*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.

```
907 \cs_new_protected:Nn \__enumext_before_args_exec_v:
908    {
909        \tl_use:N \l__enumext_before_starred_key_v_tl
910    }
```

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*⟩.

```
911 \cs_new_protected:Nn \__enumext_before_keys_exec_v:
912    {
913        \tl_use:N \l__enumext_before_no_starred_key_v_tl
914    }
```

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

```
915 \cs_new_protected:Nn \__enumext_after_stop_list_v:
916    {
917        \tl_use:N \l__enumext_after_stop_list_v_tl
918    }
```

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.

```
919 \cs_new_protected:Nn \__enumext_after_args_exec_v:
920    {
921        \tl_use:N \l__enumext_after_list_args_v_tl
922    }
```

(*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.

```
923 \cs_new_protected:Nn \__enumext_before_args_exec_vii:
924    {
925        \tl_use:N \l__enumext_before_starred_key_vii_tl
926    }
927 \cs_new_protected:Nn \__enumext_before_args_exec_viii:
928    {
929        \tl_use:N \l__enumext_before_starred_key_viii_tl
930    }
```

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*⟩.

```
931 \cs_new_protected:Nn \__enumext_before_keys_exec_vii:
932    {
933        \tl_use:N \l__enumext_before_no_starred_key_vii_tl
934    }
935 \cs_new_protected:Nn \__enumext_before_keys_exec_viii:
936    {
937        \tl_use:N \l__enumext_before_no_starred_key_viii_tl
938    }
```

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

```
939 \cs_new_protected:Nn \__enumext_after_stop_list_vii:
940    {
941        \tl_use:N \l__enumext_after_stop_list_vii_tl
942    }
943 \cs_new_protected:Nn \__enumext_after_stop_list_viii:
944    {
945        \tl_use:N \l__enumext_after_stop_list_viii_tl
946    }
```

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.

```
947 \cs_new_protected:Nn \__enumext_after_args_exec_vii:
948   {
949     \tl_use:N \l__enumext_after_list_args_vii_tl
950   }
951 \cs_new_protected:Nn \__enumext_after_args_exec_viii:
952   {
953     \tl_use:N \l__enumext_after_list_args_viii_tl
954   }
```

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

11.18   Setting keys for multicols and minipage

mini-env
mini-sep
columns-sep
columns

The default value of the columns-sep key is handled by the state of the boolean variable \l__enumext_-columns_sep_X_bool which is handled in the internal definition of the enumext and keyans environments. Define and set mini-env, mini-sep, columns-sep and columns keys for enumext, enumext*, keyans and keyans* environments.

```
955 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
956   {
957     \keys_define:nn { enumext / #1 }
958       {
959         mini-env     .dim_set:c  = { l__enumext_minipage_right_#2_dim },
960         mini-env     .value_required:n = true,
961         mini-sep     .dim_set:c  = { l__enumext_minipage_hsep_#2_dim },
962         mini-sep     .initial:n  = 0.3333em,
963         mini-sep     .value_required:n = true,
964         columns-sep .dim_set:c   = { l__enumext_columns_sep_#2_dim },
965         columns-sep .value_required:n = true,
966         columns      .int_set:c  = { l__enumext_columns_#2_int },
967         columns      .initial:n  = 1,
968         columns      .value_required:n = true,
969       }
970   }
971 \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.

```
972 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
973   {
974     \keys_define:nn { enumext / #1 }
975       {
976         mini-right  .tl_gset:c = { g__enumext_miniright_code_#2_tl },
977         mini-right  .value_required:n = true,
978         mini-right* .code:n    = {
979                                 \bool_gset_true:c { g__enumext_minipage_center_#2_bool }
980                                 \keys_set:nn { enumext / #1 } { miniright = {##1} }
981                                 },
982         mini-right* .value_required:n = true,
983       }
984   }
985 \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

`\__enumext_multi_set_vskip:`

The function `\__enumext_multi_set_vskip:` will take care of determining the *"adjusted spaces"* that we will apply *"above"* and *"below"* the `multicols` environment in enumext.

We will set the default values taking into account that TEX is in ⟨*horizontal mode*⟩, then we will make the settings for the ⟨*vertical mode*⟩ in which `\partopsep` comes into play.

Set the values of `\l__enumext_multicols_above_X_skip` and `\l__enumext_multicols_below_X_skip` equal to the value of `\topsep` in the *current level*.

```
986  \cs_new_protected:Nn \__enumext_multi_set_vskip:
987    {
988      \skip_set:cn { l__enumext_multicols_above_ \__enumext_level: _skip }
989        {
990          \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
991        }
992      \skip_set:cn { l__enumext_multicols_below_ \__enumext_level: _skip }
993        {
994          \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
995        }
996      \__enumext_add_pre_parsep:
997    }
```

(*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*.

```
998   \cs_new_protected:Nn \__enumext_add_pre_parsep:
999     {
1000      \int_case:nn { \l__enumext_level_int }
1001        {
1002          { 2 }{
1003              \skip_if_eq:nnF { \l__enumext_parsep_i_skip } { \c_zero_skip }
1004                {
1005                  \skip_add:Nn \l__enumext_multicols_above_ii_skip { \l__enumext_parsep_i_skip }
1006                }
1007            }
1008          { 3 }{
1009              \skip_if_eq:nnF { \l__enumext_parsep_ii_skip } { \c_zero_skip }
1010                {
1011                  \skip_add:Nn \l__enumext_multicols_above_iii_skip { \l__enumext_parsep_ii_skip
1012                }
1013            }
1014          { 4 }{
1015              \skip_if_eq:nnF { \l__enumext_parsep_iii_skip } { \c_zero_skip }
1016                {
1017                  \skip_add:Nn \l__enumext_multicols_above_iv_skip { \l__enumext_parsep_iii_skip
1018                }
1019            }
1020        }
1021    }
```

(*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*⟩.

```
1022  \cs_new_protected:Nn \__enumext_multi_addvspace:
1023    {
1024      \__enumext_multi_set_vskip:
1025      \mode_if_vertical:T
1026        {
```

```
1027        \skip_add:cn { l__enumext_multicols_above_ \__enumext_level: _skip }
1028          {
1029            \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
1030          }
1031        \skip_add:cn { l__enumext_multicols_below_ \__enumext_level: _skip }
1032          {
1033            \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
1034          }
1035      }
1036    \par\nopagebreak
1037    \addvspace{ \skip_use:c { l__enumext_multicols_above_ \__enumext_level: _skip } }
1038  }
```

(*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`.

```
1039  \cs_new_protected:Nn \__enumext_keyans_multi_set_vskip:
1040    {
1041      \skip_set:Nn \l__enumext_multicols_above_v_skip
1042        {
1043          \l__enumext_topsep_v_skip
1044        }
1045      \skip_set:Nn \l__enumext_multicols_below_v_skip
1046        {
1047          \l__enumext_topsep_v_skip
1048        }
1049    }
1050  \cs_new_protected:Nn \__enumext_keyans_multi_addvspace:
1051    {
1052      \__enumext_keyans_multi_set_vskip:
1053      \mode_if_vertical:T
1054        {
1055          \skip_add:Nn \l__enumext_multicols_above_v_skip
1056            {
1057              \skip_use:N \l__enumext_partopsep_v_skip
1058            }
1059          \skip_add:Nn \l__enumext_multicols_below_v_skip
1060            {
1061              \skip_use:N \l__enumext_partopsep_v_skip
1062            }
1063        }
1064      \par\nopagebreak
1065      \addvspace{ \l__enumext_multicols_above_v_skip }
1066    }
```

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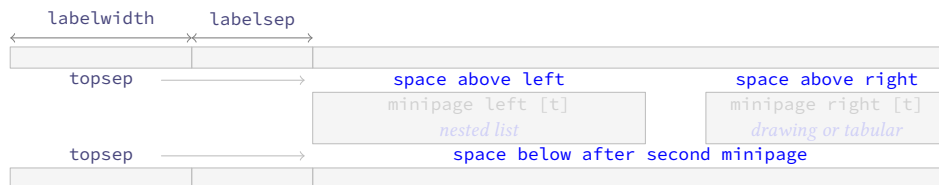
Figure 8: Representation of the `minipage` spacing adjustment for a nested level.

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.

```
1067  \cs_new_protected:Nn \__enumext_mini_set_vskip:
1068    {
1069      \int_compare:nNnTF
1070        { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } > { 1 }
1071        {
```

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.

```
1072          \skip_if_eq:nnTF
1073            { \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip } } { \c_zero_skip }
1074            {
1075              \skip_set:Nn \l__enumext_minipage_left_skip
1076                {
1077                  -0.150\box_dp:N \strutbox
1078                }
1079              \skip_set:Nn \l__enumext_minipage_right_skip
1080                {
1081                  0.695\box_dp:N \strutbox
1082                }
1083              \skip_set:Nn \l__enumext_minipage_after_skip
1084                {
1085                  \box_dp:N \strutbox
1086                }
1087              \__enumext_zero_parsep:
1088            }
1089            {
1090              \skip_set:Nn \l__enumext_minipage_left_skip
1091                {
1092                  \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
1093                }
1094              \skip_set:Nn \l__enumext_minipage_right_skip
1095                {
1096                  0.695\box_dp:N \strutbox
1097                }
1098              \skip_set:Nn \l__enumext_minipage_after_skip
1099                {
1100                  1.85\box_dp:N \strutbox
1101                  + \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
1102                }
1103            }
1104        }
1105        {
```

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.

```
1106          \skip_if_eq:nnTF
1107            { \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip } } { \c_zero_skip }
1108            {
1109              \skip_set:Nn \l__enumext_minipage_left_skip
1110                {
1111                  0.5\box_dp:N \strutbox
1112                  - \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
1113                }
1114              \skip_set:Nn \l__enumext_minipage_right_skip
```

```
1115                    {
1116                        \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
1117                    }
1118                \skip_set:Nn \l__enumext_minipage_after_skip
1119                    {
1120                        1.6\box_dp:N \strutbox
1121                    }
1122            }
1123            {
1124                \skip_set:Nn \l__enumext_minipage_left_skip
1125                    {
1126                        0.5875\box_dp:N \strutbox
1127                        - \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
1128                    }
1129                \skip_set:Nn \l__enumext_minipage_right_skip
1130                    {
1131                        + \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
1132                        + \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
1133                    }
1134                \skip_set:Nn \l__enumext_minipage_after_skip
1135                    {
1136                        0.325\box_dp:N \strutbox
1137                        + \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
1138                    }
1139            }
1140        }
1141    }
```

(*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.

```
1142  \cs_new_protected:Nn \__enumext_zero_parsep:
1143    {
1144      \int_case:nn { \l__enumext_level_int }
1145        {
1146          { 2 }{
1147                \skip_if_eq:nnF { \l__enumext_parsep_i_skip } { \c_zero_skip }
1148                  {
1149                    \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
1150                  }
1151              }
1152          { 3 }{
1153                \skip_if_eq:nnF { \l__enumext_parsep_ii_skip } { \c_zero_skip }
1154                  {
1155                    \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
1156                  }
1157              }
1158          { 4 }{
1159                \skip_if_eq:nnF { \l__enumext_parsep_iii_skip } { \c_zero_skip }
1160                  {
1161                    \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
1162                  }
1163              }
1164        }
1165    }
```

(*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*.

```
1166  \cs_new_protected:Nn \__enumext_mini_addvspace:
1167    {
1168      \__enumext_mini_set_vskip:
1169      \mode_if_vertical:T
1170        {
1171          \skip_add:Nn \l__enumext_minipage_left_skip
1172            {
```

```
1173              \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
1174            }
1175          \skip_add:Nn \l__enumext_minipage_after_skip
1176            {
1177              \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
1178            }
1179        }
1180      \par\nopagebreak
1181      \addvspace { \l__enumext_minipage_left_skip }
1182    }
```

(*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.

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

```
1237                    }
1238                \skip_set:Nn \l__enumext_minipage_after_skip
1239                    {
1240                      0.325\box_dp:N \strutbox + \l__enumext_topsep_v_skip
1241                    }
1242              }
1243          }
1244    }
```

(*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.

```
1245  \cs_new_protected:Nn \__enumext_keyans_mini_addvspace:
1246    {
1247      \__enumext_keyans_mini_set_vskip:
1248      \mode_if_vertical:T
1249        {
1250          \skip_add:Nn \l__enumext_minipage_left_skip
1251            {
1252              \l__enumext_partopsep_v_skip
1253            }
1254          \skip_add:Nn \l__enumext_minipage_after_skip
1255            {
1256              \l__enumext_partopsep_v_skip
1257            }
1258        }
1259      \par\nopagebreak
1260      \addvspace { \l__enumext_minipage_left_skip }
1261    }
```

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

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

```
1292              {
1293                  0.5\box_dp:N \strutbox
1294              }
1295          \skip_set:Nn \l__enumext_minipage_right_skip
1296              {
1297                  \l__enumext_partopsep_viii_skip
1298              }
1299          \skip_set:Nn \l__enumext_minipage_after_skip
1300              {
1301                  1.6\box_dp:N \strutbox
1302              }
1303          }
1304          {
1305          \skip_set:Nn \l__enumext_minipage_left_skip
1306              {
1307                  0.5875\box_dp:N \strutbox
1308              }
1309          \skip_set:Nn \l__enumext_minipage_right_skip
1310              {
1311                  \l__enumext_topsep_viii_skip
1312              }
1313          \skip_set:Nn \l__enumext_minipage_after_skip
1314              {
1315                  0.325\box_dp:N \strutbox + \l__enumext_topsep_viii_skip
1316              }
1317          }
1318      }
```

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

__enumext_mini_addvspace_vii:
__enumext_mini_addvspace_viii:

The functions __enumext_mini_addvspace_vii: and __enumext_mini_addvspace_viii: will apply the vertical space *"only above"* the __enumext_mini_env* environment on the *left side* when the mini-right key is active in the enumext* and keyans* environments.
Here we will NOT take into account whether TₑX is in ⟨*horizontal mode*⟩ or ⟨*vertical mode*⟩, since \partopsep is equal to 0pt in both environments.

```
1319 \cs_new_protected:Nn \__enumext_mini_addvspace_vii:
1320     {
1321         \__enumext_mini_set_vskip_vii:
1322         \par\nopagebreak
1323         \addvspace { \l__enumext_minipage_left_skip }
1324     }
1325 \cs_new_protected:Nn \__enumext_mini_addvspace_viii:
1326     {
1327         \__enumext_mini_set_vskip_viii:
1328         \par\nopagebreak
1329         \addvspace { \l__enumext_minipage_left_skip }
1330     }
```

(*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.

```
1331 \NewDocumentCommand \miniright { s }
1332     {
1333         \int_compare:nNnT { \l__enumext_keyans_pic_level_int } = { 1 }
1334             {
1335                 \msg_error:nnn { enumext } { wrong-miniright-place }
1336             }
1337         \int_compare:nNnT { \l__enumext_level_int } = { 0 }
1338             {
1339                 \msg_error:nnn { enumext } { wrong-miniright-place }
1340             }
```

```
1341     \int_compare:nNnTF { \l__enumext_keyans_level_int } = { 1 }
1342       {
1343         \__enumext_keyans_mini_right_cmd:n {#1}
1344       }
1345       { \__enumext_mini_right_cmd:n {#1} }
1346   }
```

(*End of definition for* `\miniright`*. This function is documented on page 9.*)

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

```
1347  \cs_new_protected:Npn \__enumext_mini_right_cmd:n #1
1348    {
1349      \dim_compare:nNnTF
1350        { \dim_use:c { l__enumext_minipage_right_ \__enumext_level: _dim } } > { \c_zero_dim }
1351        {
1352          \__enumext_multicols_stop:
1353          \end{__enumext_mini_env*}
1354          \hfill
1355          \begin{__enumext_mini_env*}
1356            { \dim_use:c { l__enumext_minipage_right_ \__enumext_level: _dim } }
1357            \par\addvspace { \l__enumext_minipage_right_skip }
1358            \bool_if:nF {#1}
1359              {
1360                \centering
1361              }
1362            \int_gzero:N \g__enumext_minipage_stat_int
1363        }
1364        { \msg_error:nnn { enumext } { wrong-miniright-use } }
1365    }
```

(*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.

```
1366  \cs_new_protected:Npn \__enumext_keyans_mini_right_cmd:n #1
1367    {
1368      \dim_compare:nNnTF { \l__enumext_minipage_right_v_dim } > { \c_zero_dim }
1369        {
1370          \__enumext_keyans_multicols_stop:
1371          \end{__enumext_mini_env*}
1372          \hfill
1373          \begin{__enumext_mini_env*}{ \l__enumext_minipage_right_v_dim }
1374            \par\addvspace { \l__enumext_minipage_right_skip }
1375            \bool_if:nF {#1}
1376              {
1377                \centering
1378              }
1379            \int_gzero:N \g__enumext_minipage_stat_int
1380        }
1381        { \msg_error:nnn { enumext } { wrong-miniright-use } }
1382    }
```

(*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.

```
1383 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1384   {
1385     \keys_define:nn { enumext / #1 }
1386       {
1387         above  .skip_set:c = { l__enumext_vspace_above_#2_skip },
1388         above  .value_required:n = true,
1389         above* .code:n     = \bool_set_true:c { l__enumext_vspace_a_star_#2_bool }
1390                                \keys_set:nn { enumext / #1 } { above = {##1} },
1391         above* .value_required:n = true,
1392         below  .skip_set:c = { l__enumext_vspace_below_#2_skip },
1393         below  .value_required:n = true,
1394         below* .code:n     = \bool_set_true:c { l__enumext_vspace_b_star_#2_bool }
1395                                \keys_set:nn { enumext / #1 } { below = {##1} },
1396         below* .value_required:n = true,
1397       }
1398   }
1399 \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.

```
1400 \cs_new_protected:Nn \__enumext_vspace_above:
1401   {
1402     \skip_if_eq:nnF
1403       { \skip_use:c { l__enumext_vspace_above_ \__enumext_level: _skip } } { \c_zero_skip }
1404       {
1405         \bool_if:cTF { l__enumext_vspace_a_star_ \__enumext_level: _bool }
1406           {
1407             \vspace*{ \skip_use:c { l__enumext_vspace_above_ \__enumext_level: _skip } }
1408           }
1409           {
1410             \vspace { \skip_use:c { l__enumext_vspace_above_ \__enumext_level: _skip } }
1411           }
1412       }
1413   }
```

(*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.

```
1414 \cs_new_protected:Nn \__enumext_vspace_below:
1415   {
1416     \skip_if_eq:nnF
1417       { \skip_use:c { l__enumext_vspace_below_ \__enumext_level: _skip } } { \c_zero_skip }
1418       {
1419         \bool_if:cTF { l__enumext_vspace_b_star_ \__enumext_level: _bool }
1420           {
1421             \vspace*{ \skip_use:c { l__enumext_vspace_below_ \__enumext_level: _skip } }
1422           }
1423           {
1424             \vspace { \skip_use:c { l__enumext_vspace_below_ \__enumext_level: _skip } }
1425           }
1426       }
1427   }
```

(*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.

```
1428 \cs_new_protected:Nn \__enumext_vspace_above_v:
1429   {
1430     \skip_if_eq:nnF { \l__enumext_vspace_above_v_skip } { \c_zero_skip }
1431       {
1432         \bool_if:NTF \l__enumext_vspace_a_star_v_bool
1433           {
1434             \vspace*{ \l__enumext_vspace_above_v_skip }
```

```
1435            }
1436          { \vspace { \l__enumext_vspace_above_v_skip } }
1437        }
1438    }
```

(*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.

```
1439  \cs_new_protected:Nn \__enumext_vspace_below_v:
1440    {
1441      \skip_if_eq:nnF { \l__enumext_vspace_below_v_skip } { \c_zero_skip }
1442        {
1443          \bool_if:NTF \l__enumext_vspace_b_star_v_bool
1444            {
1445              \vspace*{ \l__enumext_vspace_below_v_skip }
1446            }
1447          { \vspace { \l__enumext_vspace_below_v_skip } }
1448        }
1449    }
```

(*End of definition for* __enumext_vspace_below_v:.)

11.21.3 Functions for above and below keys in enumext* keyans*

__enumext_vspace_above_vii: The functions __enumext_vspace_above_vii: and __enumext_vspace_above_viii: apply the
__enumext_vspace_above_viii: *vertical space above* the enumext* and keyans* environments set by the above and above* keys.

```
1450  \cs_new_protected:Nn \__enumext_vspace_above_vii:
1451    {
1452      \skip_if_eq:nnF { \l__enumext_vspace_above_vii_skip } { \c_zero_skip }
1453        {
1454          \bool_if:NTF \l__enumext_vspace_a_star_vii_bool
1455            {
1456              \vspace*{ \l__enumext_vspace_above_vii_skip }
1457            }
1458          { \vspace { \l__enumext_vspace_above_vii_skip } }
1459        }
1460    }
1461  \cs_new_protected:Nn \__enumext_vspace_above_viii:
1462    {
1463      \skip_if_eq:nnF { \l__enumext_vspace_above_viii_skip } { \c_zero_skip }
1464        {
1465          \bool_if:NTF \l__enumext_vspace_a_star_viii_bool
1466            {
1467              \vspace*{ \l__enumext_vspace_above_viii_skip }
1468            }
1469          { \vspace { \l__enumext_vspace_above_viii_skip } }
1470        }
1471    }
```

(*End of definition for* __enumext_vspace_above_vii: *and* __enumext_vspace_above_viii:.)

__enumext_vspace_below_vii: The functions __enumext_vspace_below_vii: and __enumext_vspace_below_viii: apply the
__enumext_vspace_below_viii: *vertical space below* the enumext* and keyans* environments set by the below* and below keys.

```
1472  \cs_new_protected:Nn \__enumext_vspace_below_vii:
1473    {
1474      \skip_if_eq:nnF { \l__enumext_vspace_below_vii_skip } { \c_zero_skip }
1475        {
1476          \bool_if:NTF \l__enumext_vspace_b_star_vii_bool
1477            {
1478              \vspace*{ \l__enumext_vspace_below_vii_skip }
1479            }
1480          { \vspace { \l__enumext_vspace_below_vii_skip } }
1481        }
1482    }
1483  \cs_new_protected:Nn \__enumext_vspace_below_viii:
1484    {
1485      \skip_if_eq:nnF { \l__enumext_vspace_below_viii_skip } { \c_zero_skip }
1486        {
1487          \bool_if:NTF \l__enumext_vspace_b_star_viii_bool
1488            {
1489              \vspace*{ \l__enumext_vspace_below_viii_skip }
```

```
1490              }
1491            { \vspace { \l__enumext_vspace_below_viii_skip } }
1492          }
1493      }
```

(*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*.

```
1494 \cs_set_protected:Npn \__enumext_tmp:n #1
1495   {
1496     \keys_define:nn { enumext / #1 }
1497       {
1498         series  .str_set:N = \l__enumext_series_str,
1499         series  .value_required:n = true,
1500         resume  .code:n = \__enumext_resume_series:n {##1},
1501         resume* .code:n = \__enumext_resume_starred:,
1502         resume* .value_forbidden:n = true,
1503       }
1504   }
1505 \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.

```
1506 \cs_new:Npn \__enumext_filter_series:n #1
1507   {
1508     \use:e
1509       {
1510         \keyval_parse:NNn
1511           \__enumext_filter_series_key:n
1512           \__enumext_filter_series_pair:nn {#1}
1513       }
1514   }
```

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.

```
1515 \cs_new:Npn \__enumext_filter_series_key:n #1
1516   {
1517     \str_case:nnF {#1}
1518       {
1519         { resume } {}
1520         { resume* } {}
1521       }
1522       { , { \exp_not:n {#1} } }
1523   }
```

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.

```
1524 \cs_new:Npn \__enumext_filter_series_pair:nn #1#2
1525   {
1526     \str_case:nnF {#1}
1527       {
1528         { series } {}
1529         { resume } {}
1530         { start } {}
1531         { save-ans } {}
1532         { save-key } {}
1533       }
1534       { , { \exp_not:n {#1} } = { \exp_not:n {#2} } }
1535   }
```

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

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).

```
1536  \cs_new_protected:Npn \__enumext_parse_series:n #1
1537    {
1538      \str_if_empty:NTF \l__enumext_series_str
1539        {
1540          \bool_if:NF \l__enumext_resume_active_bool
1541            {
1542              \__enumext_resume_last:n {#1}
1543            }
1544        }
1545        {
1546          \tl_gclear_new:c { g__enumext_series_ \l__enumext_series_str _tl }
1547          \tl_gset:ce { g__enumext_series_ \l__enumext_series_str _tl }
1548            { \__enumext_filter_series:n {#1} }
1549          \int_if_exist:cF { g__enumext_series_ \l__enumext_series_str _int }
1550            {
1551              \int_new:c { g__enumext_series_ \l__enumext_series_str _int }
1552            }
1553        }
1554    }
```

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.

```
1555  \cs_new_protected:Npn \__enumext_resume_last:n #1
1556    {
1557      \bool_if:NT \l__enumext_standar_first_bool
1558        {
1559          \tl_gclear:N \g__enumext_standar_series_tl
1560          \tl_gset:Ne \g__enumext_standar_series_tl { \__enumext_filter_series:n {#1} }
1561        }
1562      \bool_if:NT \l__enumext_starred_first_bool
1563        {
1564          \tl_gclear:N \g__enumext_starred_series_tl
1565          \tl_gset:Ne \g__enumext_starred_series_tl { \__enumext_filter_series:n {#1} }
1566        }
1567    }
```

(*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).

```
1568  \cs_new_protected:Nn \__enumext_resume_save_counter:
1569    {
1570      \bool_if:NT \g__enumext_standar_bool
1571        {
1572          \tl_if_empty:NF \l__enumext_series_str
1573            {
1574              \int_gset_eq:cN
```

```
1575                    { g__enumext_series_ \l__enumext_series_str _int } \value{enumXi}
1576                  }
1577              \tl_if_empty:NTF \l__enumext_resume_name_tl
1578                {
1579                  \str_if_empty:NT \l__enumext_series_str
1580                    {
1581                      \int_gset_eq:NN \g__enumext_resume_int \value{enumXi}
1582                    }
1583                  }
1584                {
1585                  \int_if_exist:cT { g__enumext_series_ \l__enumext_resume_name_tl _int }
1586                    {
1587                      \int_gset_eq:cN
1588                        { g__enumext_series_ \l__enumext_resume_name_tl _int } \value{enumXi}
1589                    }
1590                  }
1591              \int_if_exist:cT { g__enumext_resume_ \l__enumext_store_name_tl _int }
1592                {
1593                  \int_gset_eq:cN
1594                    { g__enumext_resume_ \l__enumext_store_name_tl _int } \value{enumXi}
1595                }
1596          }
1597        \bool_if:NT \g__enumext_starred_bool
1598          {
1599            \tl_if_empty:NF \l__enumext_series_str
1600              {
1601                \int_gset_eq:cN
1602                  { g__enumext_series_ \l__enumext_series_str _int } \value{enumXvii}
1603              }
1604            \tl_if_empty:NTF \l__enumext_resume_name_tl
1605              {
1606                \str_if_empty:NT \l__enumext_series_str
1607                  {
1608                    \int_gset_eq:NN \g__enumext_resume_vii_int \value{enumXvii}
1609                  }
1610                }
1611              {
1612                \int_if_exist:cT { g__enumext_series_ \l__enumext_resume_name_tl _int }
1613                  {
1614                    \int_gset_eq:cN
1615                      { g__enumext_series_ \l__enumext_resume_name_tl _int } \value{enumXvii}
1616                  }
1617                }
1618            \int_if_exist:cT { g__enumext_resume_ \l__enumext_store_name_tl _int }
1619              {
1620                \int_gset_eq:cN
1621                  { g__enumext_resume_ \l__enumext_store_name_tl _int } \value{enumXvii}
1622              }
1623          }
1624      }
```

(*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.

```
1625  \cs_new_protected:Npn \__enumext_resume_series:n #1
1626    {
1627      \tl_if_empty:nTF {#1}
1628        {
1629          \__enumext_resume_counter:n { }
1630        }
1631        {
1632          \tl_if_exist:cTF { g__enumext_series_ \tl_to_str:n {#1} _tl }
1633            {
1634              \__enumext_resume_counter:n {#1}
```

```
1635          \bool_if:NT \g__enumext_standar_bool
1636            {
1637              \keys_set:nv { enumext / level-1 }
1638                { g__enumext_series_ \tl_to_str:n {#1} _tl }
1639            }
1640          \bool_if:NT \g__enumext_starred_bool
1641            {
1642              \keys_set:nv { enumext / enumext* }
1643                { g__enumext_series_ \tl_to_str:n {#1} _tl }
1644            }
1645        }
1646        {
1647          \bool_if:NT \g__enumext_standar_bool
1648            {
1649              \msg_error:nnn { enumext } { unknown-series } {#1}
1650            }
1651          \bool_if:NT \g__enumext_starred_bool
1652            {
1653              \msg_error:nnn { enumext } { unknown-series } {#1}
1654            }
1655        }
1656      }
1657    }
```

(*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.

```
1658 \cs_new_protected:Npn \__enumext_resume_counter:n #1
1659   {
1660     \bool_set_true:N \l__enumext_resume_active_bool
1661     \tl_set:Nn \l__enumext_resume_name_tl {#1}
1662     \tl_if_empty:NTF \l__enumext_resume_name_tl
1663       {
1664         \__enumext_resume_counter:
1665       }
1666       {
1667         \__enumext_resume_counter_series:
1668       }
1669     \__enumext_resume_counter_save_ans:
1670   }
```

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.

```
1671 \cs_new_protected:Nn \__enumext_resume_counter:
1672   {
1673     \bool_if:NT \g__enumext_standar_bool
1674       {
1675         \int_gincr:N \g__enumext_resume_int
1676         \int_set_eq:NN \l__enumext_start_i_int \g__enumext_resume_int
1677       }
1678     \bool_if:NT \g__enumext_starred_bool
1679       {
1680         \int_gincr:N \g__enumext_resume_vii_int
1681         \int_set_eq:NN \l__enumext_start_vii_int \g__enumext_resume_vii_int
1682       }
1683   }
```

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.

```
1684 \cs_new_protected:Nn \__enumext_resume_counter_series:
1685   {
1686     \bool_if:NT \g__enumext_standar_bool
1687       {
1688         \int_set:Nn \l__enumext_start_i_int
```

```
1689              {
1690                  \int_use:c { g__enumext_series_ \l__enumext_resume_name_tl _int } + 1
1691              }
1692          }
1693      \bool_if:NT \g__enumext_starred_bool
1694          {
1695          \int_set:Nn \l__enumext_start_vii_int
1696              {
1697                  \int_use:c { g__enumext_series_ \l__enumext_resume_name_tl _int } + 1
1698              }
1699          }
1700      }
```

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.

```
1701  \cs_new_protected:Nn \__enumext_resume_counter_save_ans:
1702    {
1703      \bool_lazy_and:nnT
1704        { \bool_if_p:N \l__enumext_standar_first_bool }
1705        { \bool_if_p:N \l__enumext_store_active_bool }
1706        {
1707          \int_set:Nn \l__enumext_start_i_int
1708            {
1709                \int_use:c { g__enumext_resume_ \l__enumext_store_name_tl _int } + 1
1710            }
1711        }
1712      \bool_lazy_and:nnT
1713        { \bool_if_p:N \l__enumext_starred_first_bool }
1714        { \bool_if_p:N \l__enumext_store_active_bool }
1715        {
1716          \int_set:Nn \l__enumext_start_vii_int
1717            {
1718                \int_use:c { g__enumext_resume_ \l__enumext_store_name_tl _int } + 1
1719            }
1720        }
1721    }
```

(*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.

```
1722  \cs_new_protected:Nn \__enumext_resume_starred:
1723    {
1724      \bool_if:NT \g__enumext_standar_bool
1725        {
1726          \tl_if_empty:NF \g__enumext_standar_series_tl
1727            {
1728                \__enumext_resume_counter:n { }
1729                \keys_set:nV { enumext / level-1 } \g__enumext_standar_series_tl
1730            }
1731        }
1732      \bool_if:NT \g__enumext_starred_bool
1733        {
1734          \tl_if_empty:NF \g__enumext_starred_series_tl
1735            {
1736                \__enumext_resume_counter:n { }
1737                \keys_set:nV { enumext / enumext* } \g__enumext_starred_series_tl
1738            }
1739        }
1740    }
```

(*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*.

```
1741 \cs_set_protected:Npn \__enumext_tmp:n #1
1742   {
1743     \keys_define:nn { enumext / #1 }
1744       {
1745         save-ans .code:n = \__enumext_storing_set:n {##1},
1746         save-ans .value_required:n = true,
1747       }
1748   }
1749 \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: The functions __enumext_start_save_ans_msg: and __enumext_stop_save_ans_msg: will dis-
__enumext_stop_save_ans_msg: play 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:.

```
1750 \cs_new_protected:Nn \__enumext_start_save_ans_msg:
1751   {
1752     \msg_term:nnVV { enumext } { save-ans-log }
1753       \g__enumext_envir_name_tl \l__enumext_store_name_tl
1754   }
1755 \cs_new_protected:Nn \__enumext_stop_save_ans_msg:
1756   {
1757     \msg_term:nnVV { enumext } { save-ans-log-hook }
1758       \g__enumext_envir_name_tl \g__enumext_store_name_tl
1759   }
```

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

__enumext_storing_set:n   The function __enumext_storing_set:n first pass the value of the save-ans key to the variable
__enumext_storing_exec: \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.

```
1760 \cs_new_protected:Npn \__enumext_storing_set:n #1
1761   {
1762     \tl_set:Ne \l__enumext_store_name_tl {#1}
1763     \tl_if_empty:NTF \l__enumext_store_name_tl
1764       {
1765         \bool_lazy_or:nnT
1766           { \l__enumext_standar_first_bool } { \l__enumext_starred_first_bool }
1767           {
1768             \msg_error:nnV { enumext } { save-ans-empty } \g__enumext_envir_name_tl
1769           }
1770       }
1771       {
1772         \bool_lazy_or:nnT
1773           { \l__enumext_standar_first_bool } { \l__enumext_starred_first_bool }
1774           {
1775             \__enumext_start_save_ans_msg:
1776             \__enumext_storing_exec:
1777           }
1778       }
1779   }
```

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 environ-
ments and will set to true the variable \l__enumext_check_answers_bool used for checking answers
by the check-ans and no-store keys, copy {⟨*store name*⟩} into the global variable \g__enumext_-
store_name_tl and execute the function __enumext_anskey_env_make:V creating the environment
anskey* (§11.27). 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*.

```
1780 \cs_new_protected:Nn \__enumext_storing_exec:
```

```
1781    {
1782      \bool_set_true:N \l__enumext_store_active_bool
1783      \bool_set_true:N \l__enumext_check_answers_bool
1784      \tl_gset:NV \g__enumext_store_name_tl \l__enumext_store_name_tl
1785      \__enumext_anskey_env_make:V \l__enumext_store_name_tl
1786      \prop_if_exist:cF { g__enumext_ \l__enumext_store_name_tl _prop }
1787        {
1788          \msg_log:nnV { enumext } { store-prop } \l__enumext_store_name_tl
1789          \prop_new:c { g__enumext_ \l__enumext_store_name_tl _prop }
1790        }
1791      \seq_if_exist:cF { g__enumext_ \l__enumext_store_name_tl _seq }
1792        {
1793          \msg_log:nnV { enumext } { store-seq } \l__enumext_store_name_tl
1794          \seq_new:c { g__enumext_ \l__enumext_store_name_tl _seq }
1795        }
1796      \int_if_exist:cF { g__enumext_resume_ \l__enumext_store_name_tl _int }
1797        {
1798          \msg_log:nnV { enumext } { store-int } \l__enumext_store_name_tl
1799          \int_new:c { g__enumext_resume_ \l__enumext_store_name_tl _int }
1800        }
1801    }
```

(*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.

```
1802  \cs_set_protected:Npn \__enumext_tmp:n #1
1803    {
1804      \keys_define:nn { enumext / #1 }
1805        {
1806          check-ans .bool_set:N = \l__enumext_check_ans_key_bool,
1807          check-ans .initial:n  = false,
1808          check-ans .value_required:n = true,
1809          no-store  .code:n = {
1810                              \bool_set_false:N \l__enumext_check_answers_bool
1811                              \bool_set_false:N \l__enumext_check_ans_key_bool
1812                              },
1813          no-store  .value_forbidden:n = true,
```

```
1814          }
1815      }
1816  \clist_map_inline:nn
1817      {
1818        level-1, level-2, level-3, level-4, enumext*
1819      }
1820      { \__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.

```
1821  \cs_new_protected:Nn \__enumext_check_ans_active:
1822      {
1823        \tl_if_empty:NF \l__enumext_store_name_tl
1824          {
1825            \bool_if:NT \l__enumext_check_answers_bool
1826              {
1827                \__enumext_check_ans_level:
1828              }
1829          }
1830      }
```

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.

```
1831  \cs_new_protected:Nn \__enumext_check_ans_level:
1832      {
1833        \int_case:nn { \l__enumext_level_int }
1834          {
1835            { 1 }{
1836                \bool_lazy_all:nT
1837                  {
1838                    { \bool_if_p:N \g__enumext_starred_bool }
1839                    { \int_compare_p:nNn { \l__enumext_level_h_int } = { 1 } }
1840                  }
1841                  {
1842                    \int_gdecr:N \g__enumext_item_number_int
1843                  }
1844              }
1845            { 2 }{
1846                \int_gdecr:N \g__enumext_item_number_int
1847              }
1848            { 3 }{
1849                \int_gdecr:N \g__enumext_item_number_int
1850              }
1851            { 4 }{
1852                \int_gdecr:N \g__enumext_item_number_int
1853              }
1854          }
```

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.

```
1855        \int_case:nn { \l__enumext_level_h_int }
1856          {
1857            { 1 }{
1858                \bool_lazy_all:nT
1859                  {
1860                    { \bool_if_p:N \g__enumext_standar_bool }
1861                    { \int_compare_p:nNn { \l__enumext_level_int } = { 1 } }
1862                  }
1863                  {
1864                    \int_gdecr:N \g__enumext_item_number_int
1865                  }
1866              }
1867          }
1868      }
```

(*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.

```
1869 \cs_new_protected:Nn \__enumext_check_ans_key_hook:
1870   {
1871     \bool_lazy_and:nnT
1872       { \bool_if_p:N \l__enumext_check_ans_key_bool }
1873       { \bool_if_p:N \g__enumext_standar_bool }
1874       {
1875         \bool_gset_true:N \g__enumext_check_ans_key_bool
1876       }
1877     \bool_lazy_and:nnT
1878       { \bool_if_p:N \l__enumext_check_ans_key_bool }
1879       { \bool_if_p:N \g__enumext_starred_bool }
1880       {
1881         \bool_gset_true:N \g__enumext_check_ans_key_bool
1882       }
1883   }
```

(*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:.

```
1884 \cs_new_protected:Nn \__enumext_item_answer_diff:
1885   {
1886     \int_gset:Nn \g__enumext_item_answer_diff_int
1887       {
1888         \int_sign:n { \g__enumext_item_number_int - \g__enumext_item_anskey_int }
1889       }
1890   }
```

(*End of definition for* __enumext_item_answer_diff:.)

__enumext_check_ans_show:
__enumext_check_ans_msg_less:
__enumext_check_ans_msg_same_ok:
__enumext_check_ans_msg_greater:

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

```
1891 \cs_new_protected:Nn \__enumext_check_ans_show:
1892   {
1893     \int_case:nn { \g__enumext_item_answer_diff_int }
1894       {
1895         { -1 }{ \__enumext_check_ans_msg_less:     }
1896         {  0 }{ \__enumext_check_ans_msg_same_ok: }
1897         {  1 }{ \__enumext_check_ans_msg_greater: }
1898       }
1899   }
1900 \cs_new_protected:Nn \__enumext_check_ans_msg_less:
1901   {
1902     \msg_warning:nneee { enumext } { item-less-answer } { \g__enumext_store_name_tl }
1903       { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
1904   }
1905 \cs_new_protected:Nn \__enumext_check_ans_msg_same_ok:
1906   {
1907     \msg_term:nneee { enumext } { items-same-answer } { \g__enumext_store_name_tl }
1908       { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
1909   }
1910 \cs_new_protected:Nn \__enumext_check_ans_msg_greater:
1911   {
1912     \msg_warning:nneee { enumext } { item-greater-answer } { \g__enumext_store_name_tl }
1913       { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
1914   }
```

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

<div style="margin-left:auto; text-align:right">

`\__enumext_check_ans_log:`
`\__enumext_check_ans_log_msg_less:`
`\__enumext_check_ans_log_msg_same_ok:`
`\__enumext_check_ans_log_msg_greater:`

</div>

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

```
1915  \cs_new_protected:Nn \__enumext_check_ans_log:
1916    {
1917      \int_case:nn { \g__enumext_item_answer_diff_int }
1918        {
1919          { -1 }{ \__enumext_check_ans_log_msg_less:     }
1920          {  0 }{ \__enumext_check_ans_log_msg_same_ok: }
1921          {  1 }{ \__enumext_check_ans_log_msg_greater: }
1922        }
1923    }
1924  \cs_new_protected:Nn \__enumext_check_ans_log_msg_less:
1925    {
1926      \msg_log:nneee { enumext } { item-less-answer } { \g__enumext_store_name_tl }
1927        { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
1928    }
1929  \cs_new_protected:Nn \__enumext_check_ans_log_msg_same_ok:
1930    {
1931      \msg_log:nneee { enumext } { items-same-answer }  { \g__enumext_store_name_tl }
1932        { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
1933    }
1934  \cs_new_protected:Nn \__enumext_check_ans_log_msg_greater:
1935    {
1936      \msg_log:nneee { enumext } { item-greater-answer } { \g__enumext_store_name_tl }
1937        { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
1938    }
```

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

11.23.6 Check for \item* and \anspic* commands

<div style="margin-left:auto; text-align:right">

`\__enumext_check_starred_cmd:n`

</div>

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.

```
1939  \cs_new_protected:Npn \__enumext_check_starred_cmd:n #1
1940    {
1941      \int_compare:nNnT
1942        { \g__enumext_check_starred_cmd_int } = { 0 }
1943        {
1944          \msg_warning:nnnV
1945            { enumext } { missing-starred }{ #1 } \l__enumext_check_start_line_env_tl
1946        }
1947      \int_compare:nNnT
1948        { \g__enumext_check_starred_cmd_int } > { 1 }
1949        {
1950          \msg_warning:nnnV
1951            { enumext } { many-starred }{ #1 } \l__enumext_check_start_line_env_tl
1952        }
1953      \int_gzero:N \g__enumext_check_starred_cmd_int
1954      \tl_clear:N \l__enumext_check_start_line_env_tl
1955    }
```

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

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

<div style="margin-left:auto; text-align:right">

`\__enumext_execute_after_env:`

</div>

The `\__enumext_execute_after_env:` function will first return the appropriate message for the end of the environment in which the save-ans key is being executed, then call the `\__enumext_item_-answer_diff:` function and then will write the values of the global variables used to the .log file. If the key check-ans is active it will execute the function `\__enumext_check_ans_show:` and show the result in the terminal, otherwise it will execute the function `\__enumext_check_ans_log:` and write the results in the .log file, undefine the environment anskey* (§11.27) 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.

```
1956  \cs_new_protected:Nn \__enumext_execute_after_env:
1957    {
1958      \int_compare:nNnT { \l__enumext_level_int } = { 0 }
1959        {
1960          \tl_if_empty:NF \g__enumext_store_name_tl
```

```
1961              {
1962                  \__enumext_stop_save_ans_msg:
1963                  \__enumext_item_answer_diff:
1964                  \__enumext_log_global_vars:
1965                  \__enumext_log_answer_vars:
1966                  \bool_if:NTF \g__enumext_check_ans_key_bool
1967                    {
1968                        \__enumext_check_ans_show:
1969                    }
1970                    { \__enumext_check_ans_log: }
1971                  \__enumext_undefine_anskey_env:
1972              }
1973            \__enumext_reset_global_vars:
1974        }
1975    }
```

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

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

```
1976 \cs_set_protected:Npn \__enumext_tmp:n #1
1977   {
1978       \keys_define:nn { enumext / #1 }
1979         {
1980             wrap-ans    .cs_set_protected:Np = \__enumext_anskey_wrapper:n ##1,
1981             wrap-ans    .initial:n = \fbox{##1},
1982             wrap-ans    .value_required:n = true,
1983             wrap-opt    .cs_set_protected:Np = \__enumext_keyans_wrapper_opt:n ##1,
1984             wrap-opt    .initial:n = [{##1}],
1985             wrap-opt    .value_required:n = true,
1986             save-sep    .tl_set:N  = \l__enumext_store_keyans_item_opt_sep_tl,
1987             save-sep    .initial:n = {, ~ },
1988             save-sep    .value_required:n = true,
1989             mark-ans    .tl_set:N  = \l__enumext_mark_answer_sym_tl,
1990             mark-ans    .initial:n = \textasteriskcentered,
1991             mark-ans    .value_required:n = true,
1992             mark-pos    .choice:,
1993             mark-pos / left     .code:n = \str_set:Nn \l__enumext_mark_position_str { l },
1994             mark-pos / right    .code:n = \str_set:Nn \l__enumext_mark_position_str { r },
1995             mark-pos / unknown .code:n =
1996                                   \msg_error:nneee { enumext } { unknown-choice }
1997                                     { mark-pos } { left, ~ right } { \exp_not:n {##1} },
1998             mark-pos    .initial:n = right,
1999             mark-pos    .value_required:n = true,
2000             show-ans    .bool_set:N = \l__enumext_show_answer_bool,
2001             show-ans    .initial:n  = false,
2002             show-ans    .value_required:n = true,
2003             show-pos    .bool_set:N = \l__enumext_show_position_bool,
2004             show-pos    .initial:n  = false,
2005             show-pos    .value_required:n = true,
2006             mark-ref    .tl_set:N   = \l__enumext_mark_ref_sym_tl,
2007             mark-ref    .initial:n  = \textasteriskcentered,
2008             mark-ref    .value_required:n = true,
2009             save-ref    .bool_set:N = \l__enumext_store_ref_key_bool,
2010             save-ref    .initial:n  = false,
2011             save-ref    .value_required:n = true,
2012         }
2013   }
2014 \clist_map_inline:nn { level-1, enumext* } { \__enumext_tmp:n {#1} }
```

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

mark-pos
show-ans
show-pos

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

```
2015 \cs_set_protected:Npn \__enumext_tmp:n #1
2016   {
```

```
2017    \keys_define:nn { enumext / #1 }
2018      {
2019        mark-pos .choice:,
2020        mark-pos / left  .code:n = \str_set:Nn \l__enumext_mark_position_str { l },
2021        mark-pos / right .code:n = \str_set:Nn \l__enumext_mark_position_str { r },
2022        mark-pos .initial:n = right,
2023        mark-pos .value_required:n  = true,
2024        show-ans .bool_set:N = \l__enumext_show_answer_bool,
2025        show-ans .initial:n  = false,
2026        show-ans .value_required:n = true,
2027        show-pos .bool_set:N = \l__enumext_show_position_bool,
2028        show-pos .initial:n  = false,
2029        show-pos .value_required:n = true,
2030      }
2031    }
2032  \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
__enumext_store_active_keys_vii:n

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

```
2033  \cs_new_protected:Npn \__enumext_store_active_keys:n #1
2034    {
2035      \bool_if:cF { l__enumext_store_save_key_ \__enumext_level: _bool }
2036        {
2037          \tl_clear:c { l__enumext_save_key_ \__enumext_level: _tl }
2038          \tl_set:ce
2039            { l__enumext_store_save_key_ \__enumext_level: _tl }
2040            { \__enumext_filter_save_key:n {#1} }
2041        }
2042    }
2043  \cs_new_protected:Npn \__enumext_store_active_keys_vii:n #1
2044    {
2045      \bool_if:NF \l__enumext_store_save_key_vii_bool
2046        {
2047          \tl_clear:N \l__enumext_store_save_key_vii_tl
2048          \tl_set:Ne \l__enumext_store_save_key_vii_tl { \__enumext_filter_save_key:n {#1} }
2049        }
2050    }
```

(*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.

```
2051  \cs_set_protected:Npn \__enumext_tmp:n #1
2052    {
2053      \keys_define:nn { enumext / enumext* }
2054        {
2055          save-key .code:n = \__enumext_parse_save_key_vii:n {##1},
2056          save-key .value_required:n = true,
2057        }
2058      \keys_define:nn { enumext / #1 }
2059        {
2060          save-key .code:n = \__enumext_parse_save_key:n {##1},
```

```
2061          save-key .value_required:n = true,
2062        }
2063   }
2064 \clist_map_inline:nn { level-1, level-2, level-3, level-4 } { \__enumext_tmp:n {#1} }
```

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

__enumext_parse_save_key:n
__enumext_parse_save_key_vii:n

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

```
2065 \cs_new_protected:Npn \__enumext_parse_save_key:n #1
2066   {
2067     \bool_set_true:c { l__enumext_store_save_key_ \__enumext_level: _bool }
2068     \tl_clear:c { l__enumext_save_key_ \__enumext_level: _tl }
2069     \tl_set:ce
2070       { l__enumext_store_save_key_ \__enumext_level: _tl }
2071       { \__enumext_filter_save_key:n {#1} }
2072   }
2073 \cs_new_protected:Npn \__enumext_parse_save_key_vii:n #1
2074   {
2075     \bool_set_true:N \l__enumext_store_save_key_vii_bool
2076     \tl_clear:N \l__enumext_store_save_key_vii_tl
2077     \tl_set:Ne \l__enumext_store_save_key_vii_tl { \__enumext_filter_save_key:n {#1} }
2078   }
```

(*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.

```
2079 \cs_new:Npn \__enumext_filter_save_key:n #1
2080   {
2081     \use:e
2082       {
2083         \keyval_parse:NNn
2084           \__enumext_filter_save_key_key:n
2085           \__enumext_filter_save_key_pair:nn {#1}
2086       }
2087   }
```

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.

```
2088 \cs_new:Npn \__enumext_filter_save_key_key:n #1
2089   {
2090     \str_case:nnF {#1}
2091       {
2092         { resume } {} { resume* } {} { no-store } {}
2093       }
2094       { , { \exp_not:n {#1} } }
2095   }
```

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.

```
2096 \cs_new:Npn \__enumext_filter_save_key_pair:nn #1#2
2097   {
2098     \str_case:nnF {#1}
2099       {
2100         { series   } {} { resume   } {} { save-ans } {}
2101         { save-ref } {} { save-key } {} { check-ans } {} { show-ans } {}
2102         { show-pos } {} { wrap-ans } {} { mark-ans  } {} { wrap-opt } {}
2103         { save-sep } {} { mark-ref } {} { mini-env  } {} { mini-sep } { }
2104         { mini-right } {} { mini-right* } {}
2105       }
2106       { , { \exp_not:n {#1} } = { \exp_not:n {#2} } }
2107   }
```

(*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.

```
2108 \cs_new_protected:Npn \__enumext_store_addto_prop:n #1
2109   {
2110     \prop_gput_if_not_in:cen { g__enumext_ \l__enumext_store_name_tl _prop }
2111       {
2112         \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } + 1 }
2113       }
2114       { #1 }
2115   }
2116 \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.

```
2117 \cs_new_protected:Npn \__enumext_store_addto_seq:n #1
2118   {
2119     \seq_gput_right:cn { g__enumext_ \l__enumext_store_name_tl _seq } { #1 }
2120   }
2121 \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.

```
2122 \cs_new_protected:Nn \__enumext_store_level_open:
2123   {
2124     \bool_if:NT \l__enumext_check_answers_bool
2125       {
2126         \tl_if_empty:cTF { l__enumext_store_save_key_ \__enumext_level: _tl }
2127           {
2128             \__enumext_store_addto_seq:n
2129               {
2130                 \item \begin{enumext}
2131               }
2132           }
2133           {
2134             \tl_put_left:cn { l__enumext_store_save_key_ \__enumext_level: _tl }
2135               {
2136                 \item \begin{enumext} [
2137               }
2138             \tl_put_right:cn { l__enumext_store_save_key_ \__enumext_level: _tl }
2139               {
2140                 ]
2141               }
2142             \__enumext_store_addto_seq:v { l__enumext_store_save_key_ \__enumext_level: _tl }
2143           }
2144       }
2145   }
2146 \cs_new_protected:Nn \__enumext_store_level_close:
2147   {
2148     \bool_if:NT \l__enumext_check_answers_bool
2149       {
2150         \__enumext_store_addto_seq:n { \end{enumext} }
2151       }
2152   }
```

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

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.

```
\__enumext_store_level_open_vii:
\__enumext_store_level_close_vii:
```

```
2153 \cs_new_protected:Nn \__enumext_store_level_open_vii:
2154   {
2155     \bool_if:NT \l__enumext_check_answers_bool
2156       {
2157         \tl_if_empty:NTF \l__enumext_store_save_key_vii_tl
2158           {
2159             \__enumext_store_addto_seq:n
2160               {
2161                 \item \mode_leave_vertical:
2162                   \vspace { -\skip_eval:n { \baselineskip + \parsep } }
2163                   \begin{enumext*}[before={\setlength{\topsep}{0pt}},]
2164               }
2165           }
2166           {
2167             \tl_put_left:Nn \l__enumext_store_save_key_vii_tl
2168               {
2169                 \item \mode_leave_vertical:
2170                   \vspace { -\skip_eval:n { \baselineskip + \parsep } }
2171                   \begin{enumext*}[before={\setlength{\topsep}{0pt}},
2172               }
2173             \tl_put_right:Nn \l__enumext_store_save_key_vii_tl
2174               {
2175                 ]
2176               }
2177             \__enumext_store_addto_seq:V \l__enumext_store_save_key_vii_tl
2178           }
2179       }
2180   }
2181 \cs_new_protected:Nn \__enumext_store_level_close_vii:
2182   {
2183     \bool_if:NT \l__enumext_check_answers_bool
2184       {
2185         \__enumext_store_addto_seq:n { \end{enumext*} }
2186       }
2187   }
```

(*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

```
2188 \cs_new_protected:Nn \__enumext_print_keyans_box:NN
2189   {
2190     \mode_leave_vertical:
2191     \skip_horizontal:n { -\dim_use:N #2 }
2192     \makebox[0pt][ r ]
2193       {
2194         \makebox[ \dim_use:N #1 ][ \l__enumext_mark_position_str ]
2195           {
2196             \tl_use:N \l__enumext_mark_answer_sym_tl
2197           }
2198       }
2199     \skip_horizontal:n { \dim_use:N #2 }
2200   }
2201 \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*.

```
2202  \keys_define:nn { enumext / anskey }
2203    {
2204      break-col .bool_set:N = \l__enumext_store_columns_break_bool,
2205      break-col .default:n  = true,
2206      break-col .value_forbidden:n = true,
2207      item-join .int_set:N = \l__enumext_store_item_join_int,
2208      item-join .value_required:n = true,
2209      item-star .bool_set:N = \l__enumext_store_item_star_bool,
2210      item-star .default:n  = true,
2211      item-star .value_forbidden:n = true,
2212      item-sym* .tl_set:N   = \l__enumext_store_item_symbol_tl,
2213      item-sym* .value_required:n = true,
2214      item-pos* .dim_set:N  = \l__enumext_store_item_symbol_sep_dim,
2215      item-pos* .value_required:n = true,
2216    }
```

💣 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.

```
2217  \NewDocumentCommand \anskey { o +m }
2218    {
2219      \__enumext_anskey_safe_outer:
2220      \group_begin:
2221        \bool_if:NT \l__enumext_check_answers_bool
2222          {
2223            \int_gincr:N \g__enumext_item_anskey_int
2224            \__enumext_anskey_safe_inner:n {#2}
2225            \__enumext_store_anskey_code:nn {#1} {#2}
2226          }
2227      \group_end:
2228    }
```

(*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.

```
2229  \cs_new_protected:Nn \__enumext_anskey_safe_outer:
2230    {
2231      \bool_if:NF \l__enumext_store_active_bool
2232        {
2233          \msg_error:nnnn { enumext } { anskey-wrong-place }{ anskey }{ enumext }
2234        }
2235      \int_compare:nNnT { \l__enumext_keyans_level_int } = { 1 }
2236        {
2237          \msg_error:nnnn { enumext } { command-wrong-place }{ anskey }{ keyans }
2238        }
2239      \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
2240        {
2241          \msg_error:nnnn { enumext } { command-wrong-place }{ anskey }{ keyans* }
2242        }
2243      \int_compare:nNnT { \l__enumext_keyans_pic_level_int } = { 1 }
2244        {
2245          \msg_error:nnnn { enumext } { command-wrong-place }{ anskey }{ keyanspic }
2246        }
2247    }
```

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.

```
2248 \cs_new_protected:Npn \__enumext_anskey_safe_inner:n #1
2249   {
2250     \tl_if_empty:nT {#1}
2251       {
2252         \msg_error:nn { enumext } { anskey-empty-arg }
2253       }
2254     \int_incr:N \l__enumext_anskey_level_int
2255     \int_compare:nNnT { \l__enumext_anskey_level_int } > { 1 }
2256       {
2257         \msg_error:nn { enumext } { anskey-nested }
2258       }
2259   }
```

(*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.

```
2260 \cs_new_protected:Npn \__enumext_store_anskey_code:nn #1 #2
2261   {
2262     \__enumext_store_addto_prop:n {#2}
2263     \bool_if:NT \l__enumext_store_ref_key_bool
2264       {
2265         \__enumext_store_internal_ref:
2266       }
2267     \__enumext_anskey_show_wrap_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`.

```
2268     \tl_if_novalue:nF {#1}
2269       {
2270         \keys_set:nn { enumext / anskey } {#1}
2271       }
2272     \tl_clear:N \l__enumext_store_anskey_arg_tl
2273     \bool_lazy_and:nnT
2274       { \bool_if_p:N \l__enumext_store_columns_break_bool }
2275       { \bool_not_p:n { \l__enumext_starred_bool } }
2276       {
2277         \tl_put_left:Nn \l__enumext_store_anskey_arg_tl { \columnbreak }
2278       }
2279     \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`.

```
2280     \bool_lazy_and:nnT
2281       { \bool_not_p:n { \l__enumext_starred_bool } }
2282       { \int_compare_p:nNn { \l__enumext_store_item_join_int } > { 1 } }
2283       {
2284         \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2285           {
2286             ( \exp_not:V \l__enumext_store_item_join_int )
2287           }
2288       }
```

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*⟩.

```
2289     \bool_if:NTF \l__enumext_store_item_star_bool
2290       {
2291         \tl_put_right:Nn \l__enumext_store_anskey_arg_tl { * }
2292         \tl_if_empty:NF \l__enumext_store_item_symbol_tl
2293           {
2294             \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2295               {
2296                 [ \exp_not:V \l__enumext_store_item_symbol_tl ]
2297               }
```

```
2298              }
2299          \dim_compare:nT
2300              {
2301                \l__enumext_store_item_symbol_sep_dim != \c_zero_dim
2302              }
2303              {
2304                \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2305                  {
2306                    [ \exp_not:V \l__enumext_store_item_symbol_sep_dim ]
2307                  }
2308              }
2309            \tl_put_right:Nn \l__enumext_store_anskey_arg_tl {#2}
2310          }
2311          {
2312            \tl_put_right:Nn \l__enumext_store_anskey_arg_tl {#2}
2313          }
```

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*⟩.

```
2314        \bool_lazy_and:nnT
2315          { \bool_if_p:N \l__enumext_store_ref_key_bool }
2316          { \bool_if_p:N \l__enumext_hyperref_bool }
2317          {
2318            \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2319              {
2320                \hfill \exp_not:N \hyperlink { \exp_not:V \l__enumext_newlabel_arg_one_tl }
2321                      { \exp_not:V \l__enumext_mark_ref_sym_tl }
2322              }
2323          }
2324        \__enumext_store_addto_seq:V \l__enumext_store_anskey_arg_tl
2325      }
```

(*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.

```
2326  \cs_new_protected:Nn \__enumext_store_internal_ref:
2327    {
2328      \cs_set_protected:Npn \__enumext_tmp:n ##1
2329        {
2330          \tl_set_eq:cc { l__enumext_label_copy_##1_tl } { l__enumext_label_##1_tl }
2331          \tl_reverse:c { l__enumext_label_copy_##1_tl }
2332          \tl_remove_once:cn { l__enumext_label_copy_##1_tl } { . }
2333          \tl_reverse:c { l__enumext_label_copy_##1_tl }
2334        }
2335      \clist_map_inline:nn { i, ii, iii, iv, vii } { \__enumext_tmp:n {##1} }
2336      \cs_set:Npn \__enumext_tmp:n ##1
2337        { . \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.

```
2338      \bool_lazy_all:nT
2339        {
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            { \tl_use:N \l__enumext_label_copy_vii_tl }
2346        }
2347      \bool_lazy_all:nT
2348        {
2349          { \bool_if_p:N \l__enumext_standar_bool }
2350          { \bool_if_p:N \g__enumext_starred_bool }
2351          { \int_compare_p:nNn { \l__enumext_level_int } > { \c_zero_int } }
2352        }
2353        {
2354          \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
```

```
2355          {
2356            \tl_use:N \l__enumext_label_copy_vii_tl
2357            \int_step_function:nnN { 1 } { \l__enumext_level_int } \__enumext_tmp:n
2358          }
2359        }
```

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.

```
2360        \bool_lazy_all:nT
2361          {
2362            { \bool_if_p:N \l__enumext_standar_bool }
2363            { \int_compare_p:nNn { \l__enumext_level_int } > { \c_zero_int } }
2364            { \int_compare_p:nNn { \l__enumext_level_h_int } = { \c_zero_int } }
2365            { \bool_not_p:n { \l__enumext_starred_bool } }
2366          }
2367          {
2368            \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2369              {
2370                \tl_use:N \l__enumext_label_copy_i_tl
2371                \int_step_function:nnN { 2 } { \l__enumext_level_int } \__enumext_tmp:n
2372              }
2373          }
2374        \cs_set:Npn \__enumext_tmp:n ##1
2375          { \tl_use:c { l__enumext_label_copy_ \int_to_roman:n {##1} _tl } }
2376        \bool_lazy_all:nT
2377          {
2378            { \bool_if_p:N \l__enumext_standar_bool }
2379            { \int_compare_p:nNn { \l__enumext_level_int } > { \c_zero_int } }
2380            { \bool_not_p:n { \g__enumext_starred_bool } }
2381            { \int_compare_p:nNn { \l__enumext_level_h_int } > { \c_zero_int } }
2382          }
2383          {
2384            \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2385              {
2386                \int_step_function:nnN { 1 } { \l__enumext_level_int } \__enumext_tmp:n
2387                . \tl_use:N \l__enumext_label_copy_vii_tl
2388              }
2389          }
```

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

```
2390        \tl_put_right:Ne \l__enumext_newlabel_arg_one_tl
2391          {
2392            \l__enumext_store_name_tl \c_colon_str
2393            \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } }
2394          }
```

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.

```
2395        \tl_put_right:Ne \l__enumext_write_aux_file_tl
2396          {
2397            \__enumext_newlabel:nn
2398              { \exp_not:V \l__enumext_newlabel_arg_one_tl }
2399              { \l__enumext_newlabel_arg_two_tl }
2400          }
2401        \l__enumext_write_aux_file_tl
2402      }
```

(*End of definition for* __enumext_store_internal_ref:.)

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

```
2403  \cs_new_protected:Npn \__enumext_anskey_show_wrap_arg:n #1
2404    {
2405      \par
2406      \bool_if:NT \l__enumext_starred_bool
2407        {
2408          \cs_set:Nn \__enumext_level: { vii }
2409        }
2410      \__enumext_print_keyans_box:cc
2411        { l__enumext_labelwidth_ \__enumext_level: _dim }
2412        { l__enumext_labelsep_ \__enumext_level: _dim }
```

```
2413        \__enumext_anskey_wrapper:n { #1 }
2414      }
```

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

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

```
2415  \cs_new_protected:Npn \__enumext_anskey_show_wrap_left:n #1
2416    {
2417      \bool_if:NT \l__enumext_show_answer_bool
2418        {
2419          \__enumext_anskey_show_wrap_arg:n { #1 }
2420        }
2421      \bool_if:NT \l__enumext_show_position_bool
2422        {
2423          \tl_set:Ne \l__enumext_mark_answer_sym_tl
2424            {
2425              \group_begin:
2426              \exp_not:N \normalfont
2427              \exp_not:N \footnotesize [ \int_eval:n
2428                {
2429                  \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
2430                }
2431                ]
2432              \group_end:
2433            }
2434          \__enumext_anskey_show_wrap_arg:n { #1 }
2435        }
2436    }
```

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

11.27 The environment anskey*

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

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

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

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

```
2437  \cs_new_protected:Nn \__enumext_undefine_anskey_env:
2438    {
2439      \cs_undefine:c { anskey* }
2440      \cs_undefine:c { endanskey* }
2441      \cs_undefine:c { __scontents_anskey*_env_begin: }
2442      \cs_undefine:c { __scontents_anskey*_env_end: }
2443    }
```

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

```
2444  \__enumext_before_env:nn { enumext }
2445    {
2446      \int_compare:nNnT { \l__enumext_level_int } = { 0 }
2447        {
2448          \cs_if_free:cF { __scontents_anskey*_env_begin: }
2449            {
2450              \msg_error:nnn { enumext } { anskey-env-error } { anskey* }
2451            }
2452        }
```

```
2453    }
2454 \__enumext_before_env:nn { enumext* }
2455    {
2456      \int_compare:nNnT { \l__enumext_level_int } = { 0 }
2457        {
2458          \cs_if_free:cF { __scontents_anskey*_env_begin: }
2459            {
2460              \msg_error:nnn { enumext } { anskey-env-error } { anskey* }
2461            }
2462        }
2463    }
```

Detection of the anskey* environment inside the keyans, keyans* and keyanspic environments.

```
2464 \__enumext_before_env:nn { anskey* }
2465    {
2466      \int_compare:nNnT { \l__enumext_keyans_level_int } = { 1 }
2467        {
2468          \msg_error:nnn { enumext } { anskey-env-wrong }{ keyans }
2469        }
2470      \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
2471        {
2472          \msg_error:nnn { enumext } { anskey-env-wrong } { keyans* }
2473        }
2474      \int_compare:nNnT { \l__enumext_keyans_pic_level_int } = { 1 }
2475        {
2476          \msg_error:nnn { enumext } { anskey-env-wrong } { keyanspic }
2477        }
2478    }
```

(*End of definition for* __enumext_undefine_anskey_env:.)

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

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

```
2479 \cs_new_protected:Npn \__enumext_anskey_env_make:n #1
2480    {
2481      \newenvsc{anskey*}[store-env=enumextanskey*#1,print-env=false]
2482      \seq_gclear_new:c
2483        { g__scontents_name_enumextanskey*#1_seq }
2484      \__enumext_anskey_env_exec:
2485    }
2486 \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.

```
2487 \cs_new_protected:Nn \__enumext_anskey_env_define_keys:
2488    {
2489      \keys_define:nn { scontents / scontents }
2490        {
2491          break-col .bool_gset:N = \g__enumext_store_columns_break_bool,
2492          break-col .default:n  = true,
2493          break-col .value_forbidden:n = true,
2494          item-join .int_gset:N = \g__enumext_store_item_join_int,
2495          item-join .value_required:n = true,
2496          item-star .bool_gset:N = \g__enumext_store_item_star_bool,
2497          item-star .default:n  = true,
2498          item-star .value_forbidden:n = true,
2499          item-sym* .tl_gset:N   = \g__enumext_store_item_symbol_tl,
2500          item-sym* .value_required:n = true,
2501          item-pos* .dim_gset:N  = \g__enumext_store_item_symbol_sep_dim,
2502          item-pos* .value_required:n = true,
2503          print-env .undefine:,
2504          store-env .undefine:,
2505          write-out .undefine:,
2506        }
2507    }
```

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.

```
2508  \cs_new_protected:Nn \__enumext_anskey_env_undefine_keys:
2509    {
2510      \keys_define:nn { scontents / scontents }
2511        {
2512          break-col .undefine:,
2513          item-join .undefine:,
2514          item-star .undefine:,
2515          item-sym* .undefine:,
2516          item-pos* .undefine:,
2517          write-out .code:n = {
2518                              \bool_set_false:N \l__scontents_storing_bool
2519                              \bool_set_true:N  \l__scontents_writing_bool
2520                              \tl_set:Nn \l__scontents_fname_out_tl {##1}
2521                            },
2522          write-out .value_required:n = true,
2523          print-env .meta:nn  = { scontents } { print-env = ##1 },
2524          print-env .default:n = true,
2525          store-env .meta:nn  = { scontents } { store-env = ##1 },
2526          unknown   .code:n   = { \__scontents_parse_environment_keys:n {##1} }
2527        }
2528    }
```

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*.

```
2529  \cs_new_protected:Npn \__enumext_rescan_anskey_env:n #1
2530    {
2531      \group_begin:
2532        \int_set:Nn \tex_newlinechar:D { `\^^J }
2533        \__scontents_rescan_tokens:x
2534          {
2535            \endgroup % This assumes \catcode`\\=0... Things might go off otherwise.
2536            #1
2537          }
2538    }
```

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

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

```
2539  \cs_new_protected:Nn \__enumext_anskey_env_exec:
2540    {
2541      \__enumext_before_env:nn { anskey* }
2542        {
2543          \__enumext_anskey_env_define_keys:
2544        }
```

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

```
2545      \__enumext_after_env:nn { anskey* }
2546        {
2547          \tl_clear:N  \l__enumext_store_anskey_env_tl
2548          \tl_clear:N  \l__enumext_store_anskey_opt_tl
2549          \seq_gpop_right:cNT
2550            { g__scontents_name_enumextanskey* \g__enumext_store_name_tl _seq }
2551            \l__enumext_store_anskey_env_tl
2552            { \seq_item:ce { g__scontents_name_enumextanskey* \g__enumext_store_name_tl _seq } { -
   1 } }
2553          \__enumext_anskey_env_keys:
2554          \__enumext_anskey_env_store:
2555          \__enumext_anskey_env_clean:
2556          \__enumext_anskey_env_undefine_keys:
2557        }
2558    }
```

🌢 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.

```
2559  \cs_new_protected:Nn \__enumext_anskey_env_keys:
2560    {
2561      \bool_lazy_and:nnT
2562        { \bool_if_p:N \g__enumext_store_columns_break_bool }
2563        { \bool_not_p:n { \l__enumext_starred_bool } }
2564        {
2565          \tl_put_left:Ne \l__enumext_store_anskey_opt_tl { ,break-col, }
2566        }
```

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

```
2567      \bool_lazy_and:nnT
2568        { \bool_not_p:n { \l__enumext_starred_bool } }
2569        { \int_compare_p:nNn { \g__enumext_store_item_join_int } > { 1 } }
2570        {
2571          \tl_put_left::Ne \l__enumext_store_anskey_opt_tl
2572            {
2573              ,item-join = \exp_not:V \g__enumext_store_item_join_int,
2574            }
2575        }
```

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

```
2576      \bool_if:NT \g__enumext_store_item_star_bool
2577        {
2578          \tl_put_left:Ne \l__enumext_store_anskey_opt_tl
2579            {
2580              ,item-star,
2581            }
2582          \tl_if_empty:NF \g__enumext_store_item_symbol_tl
2583            {
2584              \tl_put_left:Ne \l__enumext_store_anskey_opt_tl
2585                {
2586                  ,item-sym* = \exp_not:V \g__enumext_store_item_symbol_tl,
2587                }
2588            }
2589          \dim_compare:nT
2590            {
2591              \g__enumext_store_item_symbol_sep_dim != \c_zero_dim
2592            }
2593            {
2594              \tl_put_left:Ne \l__enumext_store_anskey_opt_tl
2595                {
2596                  ,item-pos* = \exp_not:V \g__enumext_store_item_symbol_sep_dim,
2597                }
2598            }
2599        }
2600    }
```

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.

```
2601  \cs_new_protected:Nn \__enumext_anskey_env_store:
2602    {
2603      \group_begin:
2604        \tl_if_empty:NF \l__enumext_store_anskey_env_tl
2605          {
2606            \tl_if_empty:NTF \l__enumext_store_anskey_opt_tl
2607              {
2608                \exp_args:Ne
2609                  \anskey
2610                    {
2611                      \__enumext_rescan_anskey_env:n { \l__enumext_store_anskey_env_tl }
2612                    }
2613              }
2614              {
```

```
2615            \keys_set:nV { enumext / anskey } \l__enumext_store_anskey_opt_tl
2616            \exp_args:Ne
2617              \anskey
2618                {
2619                  \__enumext_rescan_anskey_env:n { \l__enumext_store_anskey_env_tl }
2620                }
2621            }
2622          }
2623        \group_end:
2624      }
```

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

```
2625 \cs_new_protected:Nn \__enumext_anskey_env_clean:
2626   {
2627     \bool_gset_false:N \g__enumext_store_columns_break_bool
2628     \int_gzero:N       \g__enumext_store_item_join_int
2629     \bool_gset_false:N \g__enumext_store_item_star_bool
2630     \tl_gclear:N       \g__enumext_store_item_symbol_tl
2631     \dim_gzero:N       \g__enumext_store_item_symbol_sep_dim
2632   }
```

(*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`.

```
2633 \cs_new_protected:Npn \__enumext_keyans_addto_prop:n #1
2634   {
2635     \tl_clear:N \l__enumext_store_current_label_tl
2636     \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
2637       {
2638         \tl_put_right:Ne \l__enumext_store_current_label_tl { \l__enumext_label_vi_tl }
2639       }
2640       {
2641         \tl_put_right:Ne \l__enumext_store_current_label_tl { \l__enumext_label_v_tl }
2642       }
2643     \tl_if_novalue:nF { #1 }
2644       {
2645         % Set save-sep
2646         \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_tl
2647           {
2648             \tl_put_right:Ne \l__enumext_store_current_label_tl { \l__enumext_store_keyans_item_op
2649           }
2650         \tl_put_right:Ne \l__enumext_store_current_label_tl { #1 }
2651       }
2652     \__enumext_store_addto_prop:V \l__enumext_store_current_label_tl
2653   }
```

(*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.

```
2654 \cs_new_protected:Nn \__enumext_keyans_store_ref:
2655   {
2656     \bool_if:NT \l__enumext_store_ref_key_bool
2657       {
```

```
2658        \cs_set_protected:Npn \__enumext_tmp:n ##1
2659          {
2660            \tl_set_eq:cc { l__enumext_label_copy_##1_tl } { l__enumext_label_##1_tl }
2661            \tl_reverse:c { l__enumext_label_copy_##1_tl }
2662            \tl_remove_once:cn { l__enumext_label_copy_##1_tl } { . }
2663            \tl_reverse:c { l__enumext_label_copy_##1_tl }
2664          }
2665        \clist_map_inline:nn { i, v, vi, vii, viii } { \__enumext_tmp:n {##1} }
2666        \__enumext_keyans_store_ref_aux_i:
2667      }
2668    }
```

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.

```
2669  \cs_new_protected:Nn \__enumext_keyans_store_ref_aux_i:
2670    {
2671      \bool_if:NT \g__enumext_starred_bool
2672        {
2673          \tl_set_eq:NN \l__enumext_label_copy_i_tl \l__enumext_label_copy_vii_tl
2674        }
2675      \int_compare:nNnT { \l__enumext_keyans_pic_level_int } = { 1 }
2676        {
2677          \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2678            { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_vi_tl }
2679        }
2680      \int_compare:nNnT { \l__enumext_keyans_level_int } = { 1 }
2681        {
2682          \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2683            { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_v_tl }
2684        }
2685      \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
2686        {
2687          \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2688            { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_viii_tl }
2689        }
2690      \tl_put_right:Ne \l__enumext_newlabel_arg_one_tl
2691        {
2692          \l__enumext_store_name_tl \c_colon_str
2693          \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } }
2694        }
2695      \__enumext_keyans_store_ref_aux_ii:
2696    }
```

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.

```
2697  \cs_new_protected:Nn \__enumext_keyans_store_ref_aux_ii:
2698    {
2699      \tl_put_right:Ne \l__enumext_write_aux_file_tl
2700        {
2701          \__enumext_newlabel:nn
2702            { \exp_not:V \l__enumext_newlabel_arg_one_tl }
2703            { \l__enumext_newlabel_arg_two_tl }
2704        }
2705      \l__enumext_write_aux_file_tl
2706    }
```

(*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.

```
2707  \cs_new_protected:Npn \__enumext_keyans_addto_seq:n #1
2708    {
2709      \tl_clear:N \l__enumext_store_current_label_tl
2710      \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
2711        {
```

```
2712          \tl_put_right:Ne \l__enumext_store_current_label_tl { \item \l__enumext_label_vi_tl }
2713        }
2714        {
2715          \tl_put_right:Ne \l__enumext_store_current_label_tl { \item \l__enumext_label_v_tl }
2716        }
2717      \tl_if_novalue:nF { #1 }
2718        {
2719          \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_tl
2720            {
2721              \tl_put_right:Ne \l__enumext_store_current_label_tl
2722                {
2723                  \l__enumext_store_keyans_item_opt_sep_tl
2724                }
2725            }
2726          \tl_put_right:Ne \l__enumext_store_current_label_tl { #1 }
2727        }
2728      \__enumext_keyans_addto_seq_link:
2729    }
```

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.

```
2730 \cs_new_protected:Nn \__enumext_keyans_addto_seq_link:
2731    {
2732      \bool_lazy_and:nnT
2733        { \bool_if_p:N \l__enumext_store_ref_key_bool }
2734        { \bool_if_p:N \l__enumext_hyperref_bool }
2735        {
2736          \tl_put_right:Ne \l__enumext_store_current_label_tl
2737            {
2738              \hfill \exp_not:N \hyperlink
2739                {
2740                  \exp_not:V \l__enumext_newlabel_arg_one_tl
2741                }
2742                { \exp_not:V \l__enumext_mark_ref_sym_tl }
2743            }
2744        }
2745      \__enumext_store_addto_seq:V \l__enumext_store_current_label_tl
2746      \bool_if:NT \l__enumext_check_answers_bool
2747        {
2748          \int_gincr:N \g__enumext_item_anskey_int
2749        }
2750    }
```

(*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.

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

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

```
2751 \cs_new_protected:Npn \__enumext_keyans_show_left:n #1
2752    {
2753      \tl_if_novalue:nF { #1 }
2754        {
2755          \tl_set:Ne \l__enumext_store_current_opt_arg_tl { #1 }
2756        }
2757      \bool_if:NT \l__enumext_show_answer_bool
2758        {
2759          \__enumext_keyans_show_ans:
2760        }
2761      \bool_if:NT \l__enumext_show_position_bool
2762        {
2763          \__enumext_keyans_show_pos:
2764        }
2765    }
2766 \cs_new_protected:Nn \__enumext_keyans_show_item_opt:
```

```
2767      {
2768        \tl_if_empty:NF \l__enumext_store_current_opt_arg_tl
2769          {
2770            \bool_lazy_or:nnT
2771              { \bool_if_p:N \l__enumext_show_answer_bool }
2772              { \bool_if_p:N \l__enumext_show_position_bool }
2773              {
2774                \__enumext_keyans_wrapper_opt:n { \l__enumext_store_current_opt_arg_tl } \c_space_tl
2775              }
2776          }
2777      }
2778  \cs_new_protected:Nn \__enumext_keyans_show_ans:
2779      {
2780        \tl_put_left:Nn \l__enumext_label_v_tl
2781          {
2782            \__enumext_print_keyans_box:NN
2783              \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
2784          }
2785      }
2786  \cs_new_protected:Nn \__enumext_keyans_show_pos:
2787      {
2788        \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
2789          {
2790            \tl_set:Ne \l__enumext_mark_answer_sym_tl
2791              {
2792                \group_begin:
2793                \exp_not:N \normalfont
2794                \exp_not:N \footnotesize [ \int_eval:n
2795                  {
2796                    \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
2797                  }
2798                  ]
2799                \group_end:
2800              }
2801          }
2802          {
2803            \tl_set:Ne \l__enumext_mark_answer_sym_tl
2804              {
2805                \group_begin:
2806                \exp_not:N \normalfont
2807                \exp_not:N \footnotesize [ \int_eval:n
2808                  {
2809                    \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } + 1
2810                  }
2811                  ]
2812                \group_end:
2813              }
2814          }
2815        \tl_put_left:Nn \l__enumext_label_v_tl
2816          {
2817            \__enumext_print_keyans_box:NN
2818              \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
2819          }
2820      }
```

(*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*
```
2821  \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
2822      {
2823        \keys_define:nn { enumext / #1 }
2824          {
2825            item-sym* .tl_set:c  = { l__enumext_item_symbol_#2_tl },
2826            item-sym* .value_required:n = true,
2827            item-sym* .initial:n  = {$\star$},
2828            item-pos* .dim_set:c  = { l__enumext_item_symbol_sep_#2_dim },
2829            item-pos* .value_required:n = true,
```

```
2830              }
2831        }
2832    \clist_map_inline:nn
2833        {
2834          {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {enumext*}{vii}
2835        }
2836        { \__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.

```
2837    \cs_new_protected:Nn \__enumext_footnotetext:nn
2838        {
2839          \footnotetext[#1]{#2}
2840        }
2841    \cs_new_protected:Nn \__enumext_renew_footnote:
2842        {
2843          \seq_gclear:N \g__enumext_footnote_arg_seq
2844          \seq_gclear:N \g__enumext_footnote_int_seq
2845          \RenewDocumentCommand \footnote { o +m }
2846            {
2847              \tl_if_novalue:nTF {##1}
2848                {
2849                  \stepcounter{footnote}
2850                  \int_gset_eq:Nc \g__enumext_footnote_int { c@footnote }
2851                }
2852                {
2853                  \int_gset:Nn \g__enumext_footnote_int { ##1 }
2854                }
2855              \footnotemark [ \g__enumext_footnote_int ]
2856              \seq_gput_right:Nn \g__enumext_footnote_arg_seq { ##2 }
2857              \seq_gput_right:NV \g__enumext_footnote_int_seq \g__enumext_footnote_int
2858            }
2859        }
2860    \cs_new_protected:Nn \__enumext_print_footnote:
2861        {
2862          \seq_if_empty:NF \g__enumext_footnote_int_seq
2863            {
2864              \seq_map_pairwise_function:NNN
2865                \g__enumext_footnote_int_seq
2866                \g__enumext_footnote_arg_seq
2867                \__enumext_footnotetext:nn
2868            }
2869        }
```

(*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).

```
2870    \cs_new_protected:Npn \__enumext_default_item:n #1
2871        {
2872          \tl_if_novalue:nTF {#1}
2873            {
2874              \bool_if:NT \l__enumext_check_answers_bool
```

```
2875            {
2876              \int_gincr:N \g__enumext_item_number_int
2877            }
2878          \bool_set_true:c { l__enumext_wrap_label_ \__enumext_level: _bool }
2879          \__enumext_item_std:w \tl_use:c { l__enumext_fake_item_indent_ \__enumext_level: _tl }
2880        }
2881        {
2882          \bool_set_eq:cc
2883            { l__enumext_wrap_label_ \__enumext_level: _bool }
2884            { l__enumext_wrap_label_opt_ \__enumext_level: _bool }
2885          \__enumext_item_std:w [#1] \tl_use:c { l__enumext_fake_item_indent_ \__enumext_level: _tl
2886        }
2887    }
```

(*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).

```
2888  \cs_new_protected:Npn \__enumext_starred_item:nn #1 #2
2889    {
2890      \tl_if_novalue:nF {#1}
2891        {
2892          \tl_set:cn { l__enumext_item_symbol_ \__enumext_level: _tl } {#1}
2893        }
2894      \tl_gset_eq:Nc \g__enumext_item_symbol_tl { l__enumext_item_symbol_ \__enumext_level: _tl }
2895      \tl_if_novalue:nTF {#2}
2896        {
2897          \dim_set_eq:cc
2898            { l__enumext_item_symbol_sep_ \__enumext_level: _dim }
2899            { l__enumext_labelsep_ \__enumext_level: _dim }
2900        }
2901        {
2902          \dim_set:cn { l__enumext_item_symbol_sep_ \__enumext_level: _dim } {#2}
2903        }
2904      \bool_if:NT \l__enumext_check_answers_bool
2905        {
2906          \int_gincr:N \g__enumext_item_number_int
2907        }
2908      \bool_set_true:c { l__enumext_wrap_label_ \__enumext_level: _bool }
2909      \__enumext_item_std:w \tl_use:c { l__enumext_fake_item_indent_ \__enumext_level: _tl }
2910    }
```

(*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).

```
2911  \cs_new_protected:Nn \__enumext_redefine_item:
2912    {
2913      \RenewDocumentCommand \item { s o o }
2914        {
```

```
2915        \bool_if:nTF {##1}
2916          {
2917            \__enumext_starred_item:nn {##2} {##3}
2918          }
2919          { \__enumext_default_item:n {##2} }
2920        }
2921    }
```

(*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.

```
2922  \cs_new_protected:Npn \__enumext_keyans_default_item:n #1
2923    {
2924      \tl_if_novalue:nTF { #1 }
2925        {
2926          \bool_set_true:N \l__enumext_wrap_label_v_bool
2927          \__enumext_item_std:w \tl_use:N \l__enumext_fake_item_indent_v_tl
2928        }
2929        {
2930          \bool_set_eq:NN \l__enumext_wrap_label_v_bool \l__enumext_wrap_label_opt_v_bool
2931          \__enumext_item_std:w [#1] \tl_use:N \l__enumext_fake_item_indent_v_tl
2932        }
2933    }
```

(*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*⟩.

```
2934  \cs_new_protected:Npn \__enumext_keyans_starred_item:n #1
2935    {
2936      \tl_set_eq:NN \l__enumext_store_current_label_tmp_tl \l__enumext_label_v_tl
2937      \__enumext_keyans_show_left:n { #1 }
2938      \bool_set_true:N \l__enumext_wrap_label_v_bool
2939      \__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*⟩.

```
2940      \tl_set_eq:NN \l__enumext_label_v_tl \l__enumext_store_current_label_tmp_tl
2941      \__enumext_keyans_addto_prop:n { #1 }
2942      \__enumext_keyans_store_ref:
2943      \__enumext_keyans_addto_seq:n { #1 }
2944      \int_gincr:N \g__enumext_check_starred_cmd_int
2945    }
```

(*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).

```
2946  \cs_new_protected:Nn \__enumext_keyans_redefine_item:
2947    {
2948      \RenewDocumentCommand \item { s o }
2949        {
2950          \bool_if:nTF {##1}
2951            {
2952              \peek_remove_spaces:n
2953                {
2954                  \__enumext_keyans_starred_item:n {##2}
2955                }
2956            }
```

```
2957        {
2958            \__enumext_keyans_default_item:n {##2}
2959        }
2960     }
2961  }
```

(*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.

```
2962  \cs_new_protected:Nn \__enumext_item_starred:
2963    {
2964      \tl_if_empty:cF { l__enumext_item_symbol_ \__enumext_level: _tl }
2965        {
2966          \mode_leave_vertical:
2967          \skip_horizontal:n { -\dim_use:c { l__enumext_item_symbol_sep_ \__enumext_level: _dim } }
2968          \makebox[ 0pt ][ r ]{ \g__enumext_item_symbol_tl }
2969          \skip_horizontal:n { \dim_use:c { l__enumext_item_symbol_sep_ \__enumext_level: _dim } }
2970        }
2971    }
```

(*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).

```
2972  \cs_new_protected:Nn \__enumext_make_label:
2973    {
2974      \RenewDocumentCommand \makelabel { m }
2975        {
2976          \tl_use:c { l__enumext_label_fill_left_ \__enumext_level: _tl }
2977          \tl_use:c { l__enumext_label_font_style_ \__enumext_level: _tl }
2978          \bool_if:cTF { l__enumext_wrap_label_ \__enumext_level: _bool }
2979            {
2980              \__enumext_item_starred:
2981              \use:c { __enumext_wrapper_label_ \__enumext_level: :n } { ##1 }
2982            }
2983            { ##1 }
2984          \tl_use:c { l__enumext_label_fill_right_ \__enumext_level: _tl }
2985          \tl_gclear:N \g__enumext_item_symbol_tl
2986        }
2987    }
```

(*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).

```
2988  \cs_new_protected:Nn \__enumext_keyans_make_label:
2989    {
2990      \RenewDocumentCommand \makelabel { m }
2991        {
2992          \tl_use:N \l__enumext_label_fill_left_v_tl
2993          \tl_use:N \l__enumext_label_font_style_v_tl
2994          \bool_if:NTF \l__enumext_wrap_label_v_bool
2995            {
2996              \__enumext_wrapper_label_v:n { ##1 }
2997            }
2998            { ##1 }
2999          \tl_use:N \l__enumext_label_fill_right_v_tl
3000        }
3001    }
```

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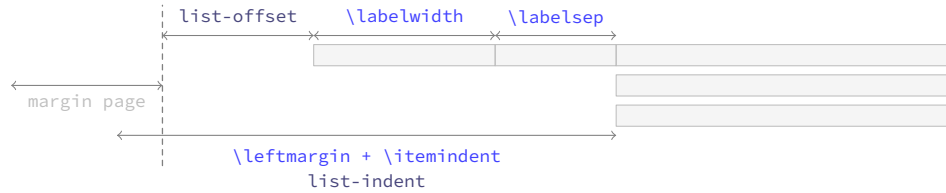
Figure 9: Representation of standard horizontal lengths in `list` environment.

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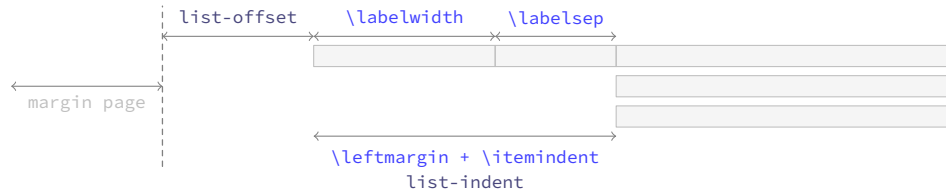
Figure 10: Representation of horizontal lengths concept in list in enumext.

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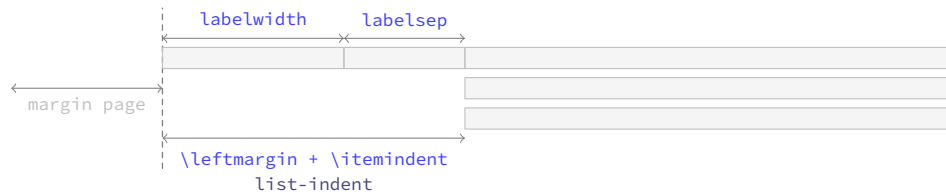
Figure 11: Default horizontal lengths in enumext.

`\__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).

```
3002  \cs_new_protected:Npn \__enumext_calc_hspace:NNNNNNN #1 #2 #3 #4 #5 #6 #7
3003    {
3004      \dim_compare:nNnT { #1 } < { \c_zero_dim }
3005        {
3006          \msg_warning:nnnV { enumext } { width-non-positive }{ labelwidth }{ #1 }
3007          \dim_set:Nn #1 { \dim_abs:n { #1 } }
3008        }
3009      \dim_compare:nNnT { #2 } < { \c_zero_dim }
3010        {
3011          \msg_warning:nnnV { enumext } { width-negative }{ labelsep }{ #2 }
3012          \dim_set:Nn #2 { \dim_abs:n { #2 } }
3013        }
```

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

```
3014      \bool_if:nF #7 { \dim_set:Nn #4 { #1 + #2} }
```

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

```
3015    \dim_compare:nNnTF { #4 } < { \c_zero_dim }
3016      {
3017        \dim_set:Nn #6 { #1 + #2 - #4}
3018        \dim_set:Nn #5 { #1 + #2 + #3 - #6 }
3019      }
3020      {
3021        \dim_compare:nNnT { #4 } = { #1 + #2 }
3022          { \dim_set:Nn #6 { \c_zero_dim } }
3023        \dim_compare:nNnT { #4 } < { #1 + #2 }
3024          { \dim_set:Nn #6 { #1 + #2 - #4} }
3025        \dim_compare:nNnT { #4 } > { #1 + #2 }
3026          {
3027            \dim_set:Nn #6 { -#1 - #2 + #4}
3028            \dim_set:Nn #6 { #6*-1}
3029          }
3030        \dim_set:Nn #5 { #1 + #2 + #3 - #6 }
3031      }
3032    }
3033  \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.

```
3034  \cs_set_protected:Npn \__enumext_tmp:n #1
3035    {
3036      \cs_new_protected:cpn { __enumext_list_arg_two_#1: }
3037        {
3038          \__enumext_calc_hspace:ccccccc
3039            { l__enumext_labelwidth_#1_dim } { l__enumext_labelsep_#1_dim }
3040            { l__enumext_listoffset_#1_dim } { l__enumext_leftmargin_tmp_#1_dim }
3041            { l__enumext_leftmargin_#1_dim } { l__enumext_itemindent_#1_dim }
3042            { l__enumext_leftmargin_tmp_#1_bool }
3043          \clist_map_inline:nn
3044            { labelsep, labelwidth, itemindent, leftmargin, rightmargin, listparindent }
3045            { \dim_set_eq:cc {####1} { l__enumext_####1_#1_dim } }
3046          \clist_map_inline:nn { topsep, parsep, partopsep, itemsep }
3047            { \skip_set_eq:cc {####1} { l__enumext_####1_#1_skip } }
3048          \usecounter { enumX#1 }
3049          \setcounter { enumX#1 } { \int_eval:n { \int_use:c { l__enumext_start_#1_int } - 1 } }
3050          \str_if_eq:nnTF {#1} { v }
3051            {
3052              \__enumext_keyans_redefine_item:
3053              \__enumext_keyans_make_label:
3054              \__enumext_keyans_ref:
3055              \__enumext_keyans_fake_item:
3056              \bool_if:cT { l__enumext_show_length_#1_bool }
3057                {
3058                  \msg_term:nnnn { enumext } { list-lengths-not-nested } { v } { keyans }
3059                }
3060            }
3061            {
3062              \__enumext_redefine_item:
3063              \__enumext_make_label:
3064              \__enumext_standar_ref:
3065              \__enumext_fake_item:
3066              \bool_if:cT { l__enumext_show_length_#1_bool }
3067                {
3068                  \msg_term:nnne { enumext } { list-lengths } {#1} { \int_use:N \l__enumext_level_i
3069                }
3070            }
3071        }
3072    }
3073  \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.

```
3074 \cs_set_protected:Npn \__enumext_tmp:n #1
3075   {
3076     \cs_new_protected:cpn { __enumext_list_arg_two_#1: }
3077       {
3078         \__enumext_calc_hspace:ccccccc
3079           { l__enumext_labelwidth_#1_dim } { l__enumext_labelsep_#1_dim }
3080           { l__enumext_listoffset_#1_dim } { l__enumext_leftmargin_tmp_#1_dim }
3081           { l__enumext_leftmargin_#1_dim } { l__enumext_itemindent_#1_dim }
3082           { l__enumext_leftmargin_tmp_#1_bool }
3083         \clist_map_inline:nn
3084           { labelsep, labelwidth, itemindent, leftmargin, rightmargin, listparindent }
3085           { \dim_set_eq:cc {####1} { l__enumext_####1_#1_dim } }
3086         \clist_map_inline:nn { topsep, parsep, partopsep, itemsep }
3087           { \skip_set_eq:cc {####1} { l__enumext_####1_#1_skip } }
3088         \skip_set_eq:Nc \parsep  { l__enumext_itemsep_#1_skip }
3089         \skip_zero:N \partopsep
3090         \usecounter { enumX#1 }
3091         \setcounter { enumX#1 } { \int_eval:n { \int_use:c { l__enumext_start_#1_int } - 1 } }
3092         \__enumext_starred_ref:
3093         \str_if_eq:nnTF {#1} { vii }
3094           {
3095             \__enumext_fake_item_vii:
3096             \bool_if:cT { l__enumext_show_length_vii_bool }
3097               { \msg_term:nnnn { enumext } { list-lengths-not-nested } { vii } { enumext* } }
3098           }
3099           {
3100             \__enumext_fake_item_viii:
3101             \bool_if:cT { l__enumext_show_length_#1_bool }
3102               { \msg_term:nnnn { enumext } { list-lengths-not-nested } { #1 } { keyans* } }
3103           }
3104       }
3105   }
3106 \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.

```
3107 \NewDocumentEnvironment{enumext}{ O{} }
3108   {
3109     \__enumext_safe_exec:
3110     \__enumext_parse_keys:n {#1}
3111     \__enumext_before_list:
3112     \__enumext_start_store_level:
3113     \__enumext_start_list:nn
3114       { \tl_use:c { l__enumext_label_ \__enumext_level: _tl } }
3115       {
3116         \use:c { __enumext_list_arg_two_ \__enumext_level: : }
3117         \__enumext_before_keys_exec:
3118       }
3119     \__enumext_after_args_exec:
3120   }
3121   {
3122     \__enumext_stop_list:
3123     \__enumext_stop_store_level:
3124     \__enumext_after_list:
3125   }
```

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

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

```
3126 \cs_new_protected:Nn \__enumext_safe_exec:
3127   {
3128
```

```
3128       \__enumext_internal_mini_page:
3129       \__enumext_is_not_nested:
3130       \int_incr:N \l__enumext_level_int
3131       \int_compare:nNnT { \l__enumext_level_int } > { 4 }
3132         { \msg_fatal:nn { enumext } { list-too-deep } }
3133       \bool_set_true:N \l__enumext_standar_bool
3134       \__enumext_is_on_first_level:
3135     }
```

(*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 environ-
ment 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.

```
3136  \cs_new_protected:Npn \__enumext_parse_keys:n #1
3137    {
3138      \tl_if_novalue:nF {#1}
3139        {
3140          \str_clear:N \l__enumext_series_str
3141          \int_compare:nNnTF { \l__enumext_level_int } = { 1 }
3142            {
3143              \keys_set:nn { enumext / level-1 } {#1}
3144              \__enumext_parse_series:n {#1}
3145            }
3146            {
3147              \exp_args:Ne \keys_set:nn
3148                { enumext / level-\int_use:N \l__enumext_level_int } {#1}
3149            }
3150          \__enumext_store_active_keys:n {#1}
3151        }
3152    }
```

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

__enumext_start_store_level:
__enumext_stop_store_level:

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

```
3153  \cs_new_protected:Nn \__enumext_start_store_level:
3154    {
3155      \bool_lazy_all:nT
3156        {
3157          { \bool_if_p:N \l__enumext_store_active_bool }
3158          { \bool_not_p:n { \l__enumext_keyans_env_bool } }
3159          { \bool_not_p:n { \g__enumext_starred_bool } }
3160        }
3161        {
3162          \int_compare:nNnT { \l__enumext_level_int } > { 1 }
3163            {
3164              \bool_set_true:c { l__enumext_store_upper_level_ \__enumext_level: _bool }
3165              \__enumext_store_level_open:
3166            }
3167        }
3168      \bool_lazy_all:nT
3169        {
3170          { \bool_if_p:N \l__enumext_store_active_bool }
3171          { \bool_not_p:n { \l__enumext_keyans_env_bool } }
3172          { \bool_if_p:N \g__enumext_starred_bool }
3173        }
3174        {
3175          \int_compare:nNnT { \l__enumext_level_int } > { 0 }
3176            {
3177              \bool_set_true:c { l__enumext_store_upper_level_ \__enumext_level: _bool }
3178              \__enumext_store_level_open:
3179            }
3180        }
3181    }
```

```
3182 \cs_new_protected:Nn \__enumext_stop_store_level:
3183   {
3184     \bool_if:cT { l__enumext_store_upper_level_ \__enumext_level: _bool }
3185       {
3186         \__enumext_store_level_close:
3187       }
3188   }
```

(*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.

```
3189 \cs_new_protected:Nn \__enumext_before_list:
3190   {
3191     \__enumext_vspace_above:
3192     \__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.

```
3193     \__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.

```
3194     \dim_compare:nNnT
3195       { \dim_use:c { l__enumext_minipage_right_ \__enumext_level: _dim } } > { \c_zero_dim }
3196       {
3197         \dim_set:cn { l__enumext_minipage_left_ \__enumext_level: _dim }
3198           {
3199             \linewidth
3200             - \dim_use:c { l__enumext_minipage_right_ \__enumext_level: _dim }
3201             - \dim_use:c { l__enumext_minipage_hsep_ \__enumext_level: _dim }
3202           }
```

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*.

```
3203         \bool_set_true:c { l__enumext_minipage_active_ \__enumext_level: _bool }
3204         \int_gincr:N \g__enumext_minipage_stat_int
3205         \__enumext_mini_addvspace:
3206         \nointerlineskip\noindent
3207         \begin{__enumext_mini_env*}
3208           { \dim_use:c { l__enumext_minipage_left_ \__enumext_level: _dim } }
3209       }
3210     \__enumext_multicols_start:
3211   }
```

(*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.

```
3212 \cs_new_protected:Nn \__enumext_multicols_start:
3213   {
3214     \int_compare:nNnT
3215       { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } > { 1 }
3216       {
3217         \dim_compare:nNnT
3218           { \dim_use:c { l__enumext_columns_sep_ \__enumext_level: _dim } } = { \c_zero_dim }
3219           {
3220             \dim_set:cn { l__enumext_columns_sep_ \__enumext_level: _dim }
3221               {
3222                 ( \dim_use:c { l__enumext_labelwidth_ \__enumext_level: _dim }
```

```
3223                    + \dim_use:c { l__enumext_labelsep_ \__enumext_level: _dim }
3224                  ) / \int_use:c { l__enumext_columns_ \__enumext_level: _int }
3225                  - \dim_use:c { l__enumext_listoffset_ \__enumext_level: _dim }
3226                }
3227            }
3228          \dim_set_eq:Nc \columnsep { l__enumext_columns_sep_ \__enumext_level: _dim  }
3229          \skip_zero:N \multicolsep
3230          \int_compare:nNnT { \l__enumext_level_int } > { 1 }
3231            {
3232              \dim_zero:N \columnseprule
3233            }
```

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.

```
3234          \bool_if:cF { l__enumext_minipage_active_ \__enumext_level: _bool }
3235            {
3236              \__enumext_multi_addvspace:
3237            }
3238          \raggedcolumns
3239          \begin{multicols}{ \int_use:c { l__enumext_columns_ \__enumext_level: _int } } }
3240        }
3241    }
```

(*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.

```
3242  \cs_new_protected:Nn \__enumext_multicols_stop:
3243    {
3244      \int_compare:nNnT
3245        { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } > { 1 }
3246        {
3247          \end{multicols}
3248          \bool_if:cF { l__enumext_minipage_active_ \__enumext_level: _bool }
3249            {
3250              \par\addvspace{ \skip_use:c { l__enumext_multicols_below_ \__enumext_level: _skip } }
3251            }
3252        }
3253    }
```

(*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.

```
3254  \cs_new_protected:Nn \__enumext_after_list:
3255    {
3256      \bool_if:cTF { l__enumext_minipage_active_ \__enumext_level: _bool }
3257        {
3258          \int_compare:nNnT { \g__enumext_minipage_stat_int } = { 1 }
3259            {
3260              \msg_warning:nn { enumext } { missing-miniright }
3261              \miniright
3262            }
3263          \int_gzero:N \g__enumext_minipage_stat_int
3264          \end{__enumext_mini_env*}
3265          \par\addvspace { \l__enumext_minipage_after_skip }
3266        }
3267        { \__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.

```
3268        \__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.

```
3269     \__enumext_after_stop_list:
3270     \__enumext_vspace_below:
3271     \bool_set_false:N \l__enumext_standar_bool
3272     \__enumext_resume_save_counter:
3273   }
```

(*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.

```
3274 \__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.

```
3275 \NewDocumentEnvironment{keyans}{ O{} }
3276   {
3277     \__enumext_keyans_safe_exec:
3278     \__enumext_keyans_parse_keys:n {#1}
3279     \__enumext_before_list_v:
3280     \__enumext_start_list:nn
3281       { \tl_use:N \l__enumext_label_v_tl }
3282       {
3283         \__enumext_list_arg_two_v:
3284         \__enumext_before_keys_exec_v:
3285       }
3286     \__enumext_after_args_exec_v:
3287   }
3288   {
3289     \__enumext_check_starred_cmd:n { item }
3290     \__enumext_stop_list:
3291     \__enumext_after_list_v:
3292   }
```

(*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.

```
3293 \cs_new_protected:Nn \__enumext_keyans_safe_exec:
3294   {
3295     \bool_if:NF \l__enumext_store_active_bool
3296       {
3297         \msg_error:nnnn { enumext } { wrong-place }{ keyans }{ save-ans }
3298       }
3299     \int_incr:N \l__enumext_keyans_level_int
3300     \bool_set_true:N \l__enumext_keyans_env_bool
3301     \__enumext_keyans_start_line:
3302     % Set false for interfering with enumext nested in keyans (yes, its possible and crayze)
3303     \bool_set_false:N \l__enumext_store_active_bool
3304     \int_compare:nNnT { \l__enumext_keyans_level_int } > { 1 }
3305       {
3306         \msg_error:nn { enumext } { keyans-nested }
3307       }
3308     \int_compare:nNnT { \l__enumext_level_int } > { 1 }
3309       {
3310         \msg_error:nn { enumext } { keyans-wrong-level }
3311       }
3312   }
```

(*End of definition for* __enumext_keyans_safe_exec:.)

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

```
3313 \cs_new_protected:Npn \__enumext_keyans_parse_keys:n #1
3314   {
3315     \keys_set:nn { enumext / keyans } {#1}
3316   }
```

(*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.

```
3317 \cs_new_protected:Nn \__enumext_before_list_v:
3318   {
3319     \__enumext_vspace_above_v:
3320     \__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.

```
3321     \dim_compare:nNnT { \l__enumext_minipage_right_v_dim } > { \c_zero_dim }
3322       {
3323         \dim_set:Nn \l__enumext_minipage_left_v_dim
3324           {
3325             \linewidth - \l__enumext_minipage_right_v_dim - \l__enumext_minipage_hsep_v_dim
3326           }
```

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*.

```
3327         \bool_set_true:N \l__enumext_minipage_active_v_bool
3328         \int_gincr:N \g__enumext_minipage_stat_int
3329         \__enumext_keyans_mini_addvspace:
3330         \nointerlineskip\noindent
3331         \begin{__enumext_mini_env*}{ \l__enumext_minipage_left_v_dim }
3332       }
```

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

```
3333     \__enumext_keyans_multicols_start:
3334   }
```

(*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.

```
3335 \cs_new_protected:Nn \__enumext_keyans_multicols_start:
3336   {
3337     \int_compare:nNnT { \l__enumext_columns_v_int } > { 1 }
3338       {
```

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

```
3339         \dim_compare:nNnT { \l__enumext_columns_sep_v_dim } = { \c_zero_dim }
3340           {
3341             \dim_set:Nn \l__enumext_columns_sep_v_dim
3342               {
3343                 (
3344                   \l__enumext_labelwidth_v_dim + \l__enumext_labelsep_v_dim
3345                 ) / \l__enumext_columns_v_int
3346                 - \l__enumext_listoffset_v_dim
3347               }
3348           }
3349         \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).

```
3350         \skip_zero:N \multicolsep
3351         \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.

```
3352         \bool_if:NF \l__enumext_minipage_active_v_bool
3353           {
3354             \__enumext_keyans_multi_addvspace:
```

```
3355            }
3356          \raggedcolumns
3357          \begin{multicols}{ \l__enumext_columns_v_int }
3358        }
3359    }
```

(*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.

```
3360  \cs_new_protected:Nn \__enumext_keyans_multicols_stop:
3361    {
3362      \int_compare:nNnT { \l__enumext_columns_v_int } > { 1 }
3363        {
3364          \end{multicols}
3365          \bool_if:NF \l__enumext_minipage_active_v_bool
3366            {
3367              \par\addvspace{ \l__enumext_multicols_below_v_skip }
3368            }
3369        }
3370    }
```

(*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.

```
3371  \cs_new_protected:Nn \__enumext_after_list_v:
3372    {
3373      \bool_if:NTF \l__enumext_minipage_active_v_bool
3374        {
3375          \int_compare:nNnT { \g__enumext_minipage_stat_int } = { 1 }
3376            {
3377              \msg_warning:nn { enumext } { missing-miniright }
3378              \miniright
3379            }
3380          \int_gzero:N \g__enumext_minipage_stat_int
3381          \end{__enumext_mini_env*}
3382          \par\addvspace{ \l__enumext_minipage_after_skip }
3383        }
3384        { \__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.

```
3385      \bool_set_false:N \l__enumext_keyans_env_bool
3386      \__enumext_after_stop_list_v:
3387      \__enumext_vspace_below_v:
3388    }
```

(*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.

This implementation is adapted from the answer given by Enrico Gregorio in How to process the body of an environment and divide it by a \macro?.
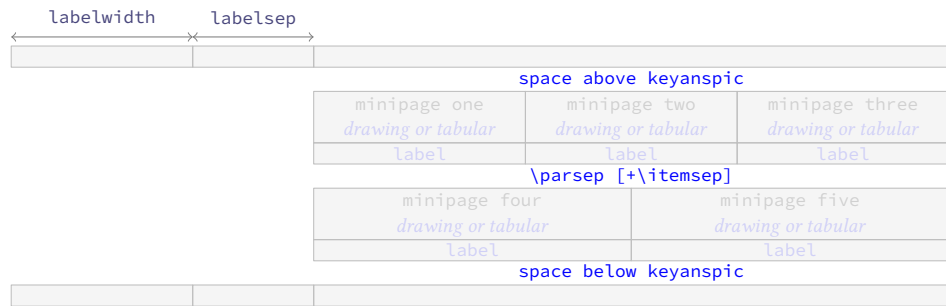
Figure 12: Representation of the keyanspic spacing in enumext.

11.36.1 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.

```
3389  \NewDocumentCommand \anspic { s o +m }
3390    {
```

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

```
3391      \bool_if:NF \l__enumext_store_active_bool
3392        {
3393          \msg_error:nnnn { enumext } { wrong-place }{ keyanspic }{ save-ans }
3394        }
3395      \int_compare:nNnT { \l__enumext_level_int } > { 1 }
3396        {
3397          \msg_error:nn { enumext } { keyanspic-wrong-level }
3398        }
3399      \int_compare:nNnT { \l__enumext_keyans_level_int } = { 1 }
3400        {
3401          \msg_error:nnnn { enumext } { command-wrong-place }{ anspic }{ keyans }
3402        }
```

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.

```
3403      \seq_put_right:Nn \l__enumext_keyans_pic_body_seq
3404        {
3405          \__enumext_keyans_anspic_code:nnn { #1 } { #2 } { #3 }
3406        }
3407    }
```

(*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.

```
3408  \cs_new_protected:Nn \__enumext_keyans_anspic_code:nnn
3409    {
3410      \stepcounter { enumXvi }
3411      #3 \\
3412      \bool_if:nT { #1 }
3413        {
3414          \__enumext_keyans_addto_prop:n { #2 }
3415          \__enumext_keyans_store_ref:
3416          \__enumext_keyans_addto_seq:n { #2 }
3417          \int_gincr:N \g__enumext_check_starred_cmd_int
3418          \bool_lazy_or:nnT
3419            { \bool_if_p:N \l__enumext_show_answer_bool }
3420            { \bool_if_p:N \l__enumext_show_position_bool }
3421            {
3422              \tl_set_eq:NN \l__enumext_label_v_tl \l__enumext_label_vi_tl
3423              \__enumext_keyans_show_left:n { #2 }
3424              \tl_set_eq:NN \l__enumext_label_vi_tl \l__enumext_label_v_tl
3425            }
3426        }
3427      \tl_use:N \l__enumext_label_font_style_v_tl
3428      \__enumext_wrapper_label_v:n { \l__enumext_label_vi_tl } \__enumext_keyans_show_item_opt:
3429    }
```

(*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.

```
3430  \NewDocumentEnvironment{keyanspic}{ o }
3431    {
3432      \__enumext_keyans_pic_safe_exec:
3433      \__enumext_start_list:nn
3434        { }
3435        {
3436          \__enumext_keyans_pic_arg_two:
3437        }
```

We apply the "adjusted" vertical spacing above the environment

```
3438        \vspace { \l__enumext_keyans_pic_above_skip }
3439    }
```

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.

```
3440    {
3441      \tl_if_novalue:nTF { #1 }
3442        {
3443          \__enumext_keyans_pic_do:e { \seq_count:N \l__enumext_keyans_pic_body_seq }
3444        }
3445        { \__enumext_keyans_pic_do:n { #1 } }
3446      \__enumext_stop_list:
3447      \__enumext_check_starred_cmd:n { anspic }
3448      \setcounter { enumXvi } { 0 }
3449      \vspace { \l__enumext_topsep_v_skip }
3450      %\bool_set_false:N \l__enumext_store_active_bool
3451    }
```

(*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.

```
3452  \cs_new_protected:Nn \__enumext_keyans_pic_safe_exec:
3453    {
3454      \int_incr:N \l__enumext_keyans_pic_level_int
3455      \int_compare:nNnT { \l__enumext_keyans_pic_level_int } > { 1 }
3456        {
3457          \msg_error:nn { enumext } { keyanspic-nested }
3458        }
3459      \__enumext_keyans_start_line:
3460    }
```

(*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.

```
3461  \cs_new_protected:Npn \__enumext_keyans_pic_skip_abs:N #1
3462    {
3463      \dim_compare:nNnT { #1 } < { 0pt }
3464        { \skip_set:Nn #1 { -#1 } }
3465    }
```

(*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.

```
3466  \cs_new_protected:Nn \__enumext_keyans_pic_arg_two:
3467    {
```

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.

```
3468      \bool_set_false:N \l__enumext_leftmargin_tmp_v_bool
3469      \__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.

```
3470    \skip_add:Nn \parsep { \itemsep }
3471    \dim_add:Nn  \leftmargin { -\labelwidth - \labelsep }
3472    \dim_zero:N  \labelwidth
3473    \dim_zero:N  \listparindent
3474    \dim_zero:N  \labelsep
3475    \skip_zero:N \partopsep
3476    \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.

```
3477    \__enumext_keyans_pic_skip_abs:N \parsep
3478    \skip_set:Nn \l__enumext_keyans_pic_above_skip
3479      {
3480        \box_dp:N \strutbox
3481        + \l__enumext_topsep_v_skip
3482        - \parsep
3483      }
3484    \__enumext_item_std:w \scan_stop:
3485  }
```

(*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`.

```
3486  \cs_new_protected:Nn \__enumext_keyans_pic_do:n
3487    {
3488      \clist_map_function:nN { #1 } \__enumext_keyans_pic_row:n
3489    }
3490  \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.

```
3491  \cs_new_protected:Nn \__enumext_keyans_pic_row:n
3492    {
3493      \dim_set:Nn \l__enumext_keyans_pic_width_dim { \linewidth / #1 }
3494      \int_set:Nn \l__enumext_keyans_pic_above_int { \l__enumext_keyans_pic_below_int }
3495      \int_set:Nn \l__enumext_keyans_pic_below_int { \l__enumext_keyans_pic_above_int + #1 }
3496      \int_step_inline:nnn
3497        { \l__enumext_keyans_pic_above_int + 1 }
3498        { \l__enumext_keyans_pic_below_int }
3499        {
3500          \__enumext_minipage:w [ b ]{ \l__enumext_keyans_pic_width_dim }
3501            \centering
3502            \seq_item:Nn \l__enumext_keyans_pic_body_seq { ##1 }
3503          \__enumext_endminipage:
3504        }
3505      \par
3506    }
```

(*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.

```
3507  \cs_new_protected:Nn \__enumext_starred_columns_set_vii:
3508    {
3509      \dim_compare:nNnT { \l__enumext_columns_sep_vii_dim } = { \c_zero_dim }
3510        {
3511          \dim_set:Nn \l__enumext_columns_sep_vii_dim
3512            {
3513              ( \l__enumext_labelwidth_vii_dim + \l__enumext_labelsep_vii_dim )
3514              / \l__enumext_columns_vii_int
3515            }
3516        }
3517      \int_set:Nn \l__enumext_tmpa_vii_int { \l__enumext_columns_vii_int - \c_one_int }
3518      \dim_set:Nn \l__enumext_item_width_vii_dim
3519        {
3520          ( \linewidth - \l__enumext_columns_sep_vii_dim * \l__enumext_tmpa_vii_int )
3521          / \l__enumext_columns_vii_int - \l__enumext_labelwidth_vii_dim
3522          - \l__enumext_labelsep_vii_dim
3523        }
3524      \dim_zero_new:N \itemwidth
3525    }
```

(*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`.

```
3526  \cs_new_protected:Npn \__enumext_starred_joined_item_vii:n #1
3527    {
3528      \int_set:Nn \l__enumext_joined_item_vii_int {#1}
3529      \int_compare:nNnT { \l__enumext_joined_item_vii_int } > { \l__enumext_columns_vii_int }
3530        {
3531          \msg_warning:nnee { enumext } { item-joined }
3532            { \int_use:N \l__enumext_joined_item_vii_int }
3533            { \int_use:N \l__enumext_columns_vii_int }
3534          \int_set:Nn \l__enumext_joined_item_vii_int
3535            {
3536              \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + \c_one_int
3537            }
3538        }
3539      \int_compare:nNnT
3540        { \l__enumext_joined_item_vii_int }
3541          >
3542        { \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + \c_one_int }
3543        {
3544          \msg_warning:nnee { enumext } { item-joined-columns }
3545            { \int_use:N \l__enumext_joined_item_vii_int }
3546            {
3547              \int_eval:n
3548                { \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + \c_one_int }
3549            }
3550          \int_set:Nn \l__enumext_joined_item_vii_int
3551            {
3552              \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + \c_one_int
3553            }
3554        }
```

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

```
3555        \int_compare:nNnTF { \l__enumext_joined_item_vii_int } > { \c_one_int }
3556          {
3557            \int_set_eq:NN \l__enumext_joined_item_aux_vii_int \l__enumext_joined_item_vii_int
3558            \int_decr:N \l__enumext_joined_item_aux_vii_int
3559            \int_add:Nn \l__enumext_item_column_pos_vii_int { \l__enumext_joined_item_aux_vii_int }
3560            \int_gadd:Nn \g__enumext_item_count_all_vii_int { \l__enumext_joined_item_aux_vii_int }
3561            \dim_set:Nn \l__enumext_joined_width_vii_dim
3562              {
3563                \l__enumext_item_width_vii_dim * \l__enumext_joined_item_vii_int
3564                + (  \l__enumext_labelwidth_vii_dim + \l__enumext_labelsep_vii_dim
3565                  + \l__enumext_columns_sep_vii_dim
3566                )*\l__enumext_joined_item_aux_vii_int
```

```
3567                }
3568            \dim_set_eq:NN \itemwidth \l__enumext_joined_width_vii_dim
3569        }
3570        {
3571            \dim_set_eq:NN \l__enumext_joined_width_vii_dim \l__enumext_item_width_vii_dim
3572            \dim_set_eq:NN \itemwidth \l__enumext_item_width_vii_dim
3573        }
3574    }
```

(*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.

```
3575 \cs_new_protected:Nn \__enumext_start_mini_vii:
3576    {
3577        \dim_compare:nNnT { \l__enumext_minipage_right_vii_dim } > { \c_zero_dim }
3578        {
3579            \dim_set:Nn \l__enumext_minipage_left_vii_dim
3580            {
3581                \linewidth
3582                - \l__enumext_minipage_right_vii_dim
3583                - \l__enumext_minipage_hsep_vii_dim
3584            }
3585            \bool_set_true:N \l__enumext_minipage_active_vii_bool
3586            \dim_gset_eq:NN
3587                \g__enumext_minipage_right_vii_dim
3588                \l__enumext_minipage_right_vii_dim
3589            \__enumext_mini_addvspace_vii:
3590            \nointerlineskip\noindent
3591            \begin{__enumext_mini_env*}{ \l__enumext_minipage_left_vii_dim }
3592        }
3593    }
```

(*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"*.

```
3594 \cs_new_protected:Nn \__enumext_stop_mini_vii:
3595    {
3596        \bool_if:NT \l__enumext_minipage_active_vii_bool
3597        {
3598            \end{__enumext_mini_env*}
3599            \hfill
3600            \bool_gset_true:N \g__enumext_minipage_active_vii_bool
3601        }
3602    }
```

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"*.

```
3603 \__enumext_after_env:nn {enumext*}
3604    {
3605        \bool_if:NT \g__enumext_minipage_active_vii_bool
3606        {
3607            \begin{__enumext_mini_env*}{ \g__enumext_minipage_right_vii_dim }
3608                \par\addvspace { \g__enumext_minipage_right_skip }
3609                \bool_if:NF \g__enumext_minipage_center_vii_bool
3610                {
3611                    \centering
3612                }
3613                \tl_use:N \g__enumext_miniright_code_vii_tl % the code
3614            \end{__enumext_mini_env*}
3615            \par\addvspace{ \g__enumext_minipage_after_skip }
3616        }
3617        \bool_gset_false:N \g__enumext_minipage_active_vii_bool
3618        \bool_gset_true:N \g__enumext_minipage_center_vii_bool
3619        \tl_gclear:N \g__enumext_miniright_code_vii_tl
3620        \dim_gzero:N \g__enumext_minipage_right_vii_dim
```

```
3621      \bool_gset_false:N \g__enumext_starred_bool
3622    }
```

(*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.

```
3623  \NewDocumentEnvironment{enumext*}{ o }
3624    {
3625      \__enumext_safe_exec_vii:
3626      \__enumext_parse_keys_vii:n {#1}
3627      \__enumext_before_list_vii:
3628      \__enumext_start_store_level_vii:
3629      \__enumext_start_list:nn { }
3630        {
3631          \__enumext_list_arg_two_vii:
3632          \__enumext_before_keys_exec_vii:
3633        }
3634      \__enumext_starred_columns_set_vii:
3635      \item[] \scan_stop:
3636      \cs_set_eq:NN \__enumext_stop_item_tmp_vii: \noindent
3637      \cs_set_eq:NN \item \__enumext_start_item_tmp_vii:
3638    }
3639    {
3640      \__enumext_stop_item_tmp_vii:
3641      \__enumext_remove_extra_parsep_vii:
3642      \__enumext_stop_list:
3643      \__enumext_stop_store_level_vii:
3644      \__enumext_after_list_vii:
3645    }
```

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

__enumext_safe_exec_vii: First check the maximum nesting level for the enumext* environment then set the vars \l__enumext_-starred_bool and \g__enumext_starred_bool.

```
3646  \cs_new_protected:Nn \__enumext_safe_exec_vii:
3647    {
3648      \__enumext_internal_mini_page:
3649      \__enumext_is_not_nested:
3650      \int_incr:N \l__enumext_level_h_int
3651      \int_compare:nNnT { \l__enumext_level_h_int } > { 1 }
3652        {
3653          \msg_error:nn { enumext } { nested }
3654        }
3655      \bool_set_true:N \l__enumext_starred_bool
3656      \__enumext_is_on_first_level:
3657    }
```

(*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*⟩.

```
3658  \cs_new_protected:Npn \__enumext_parse_keys_vii:n #1
3659    {
3660      \tl_if_novalue:nF {#1}
3661        {
3662          \str_clear:N \l__enumext_series_str
3663          \keys_set:nn { enumext / enumext* } {#1}
3664          \__enumext_parse_series:n {#1}
3665          \__enumext_store_active_keys_vii:n {#1}
3666        }
3667    }
```

(*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.

```
3668  \cs_new_protected:Nn \__enumext_before_list_vii:
3669    {
3670      \__enumext_vspace_above_vii:
3671      \__enumext_check_ans_active:
3672      \__enumext_before_args_exec_vii:
3673      \__enumext_start_mini_vii:
3674    }
```

(*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.

```
3675  \cs_new_protected:Nn \__enumext_after_list_vii:
3676    {
3677      \__enumext_stop_mini_vii:
3678      \__enumext_after_stop_list_vii:
3679      \__enumext_check_ans_key_hook:
3680      \__enumext_vspace_below_vii:
3681      \bool_set_false:N \l__enumext_starred_bool
3682      \__enumext_resume_save_counter:
3683    }
```

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

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

```
3684  \cs_new_protected:Nn \__enumext_start_store_level_vii:
3685    {
3686      \bool_if:NT \l__enumext_store_active_bool
3687        {
3688          \int_compare:nNnT { \l__enumext_level_int } > { \c_zero_int }
3689            {
3690              \__enumext_store_level_open_vii:
3691            }
3692        }
3693    }
3694  \cs_new_protected:Nn \__enumext_stop_store_level_vii:
3695    {
3696      \bool_if:NT \l__enumext_store_active_bool
3697        {
3698          \int_compare:nNnT { \l__enumext_level_int } > { \c_zero_int }
3699            {
3700              \__enumext_store_level_close_vii:
3701            }
3702        }
3703    }
```

(*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.

```
3704  \cs_new_protected_nopar:Nn \__enumext_start_item_tmp_vii:
3705    {
3706      \__enumext_stop_item_tmp_vii:
3707      \int_incr:N \l__enumext_item_column_pos_vii_int
3708      \int_gincr:N \g__enumext_item_count_all_vii_int
3709      \__enumext_item_peek_args_vii:
3710    }
```

(*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).

```
3711 \cs_new_protected:Nn \__enumext_item_peek_args_vii:
3712   {
3713     \peek_meaning:NTF (
3714       { \__enumext_joined_item_vii:w }
3715       { \__enumext_joined_item_vii:w (1) }
3716   }
```

(*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.

```
3717 \cs_new_protected:Npn \__enumext_joined_item_vii:w (#1)
3718   {
3719     \__enumext_starred_joined_item_vii:n {#1}
3720     \peek_meaning_remove:NTF *
3721       { \__enumext_starred_item_vii:w  }
3722       { \__enumext_standar_item_vii:w }
3723   }
```

(*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].

```
3724 \cs_new_protected:Npn \__enumext_standar_item_vii:w
3725   {
3726     \bool_set_false:N \l__enumext_item_starred_vii_bool
3727     \peek_meaning:NTF [
3728       {
3729         \bool_set_eq:NN
3730           \l__enumext_wrap_label_vii_bool
3731           \l__enumext_wrap_label_opt_vii_bool
3732         \__enumext_start_item_vii:w
3733       }
3734       {
3735         \bool_set_true:N \l__enumext_wrap_label_vii_bool
3736         \legacy_if_set_true:n { @noitemarg }
3737         \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
3738       }
3739   }
```

(*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*⟩].

```
3740 \cs_new_protected:Npn \__enumext_starred_item_vii:w
3741   {
3742     \bool_set_true:N \l__enumext_item_starred_vii_bool
3743     \bool_set_true:N \l__enumext_wrap_label_vii_bool
3744     \peek_meaning:NTF [
3745       { \__enumext_starred_item_vii_aux_i:w }
3746       { \__enumext_starred_item_vii_aux_ii:w }
3747   }
3748 \cs_new_protected:Npn \__enumext_starred_item_vii_aux_i:w [#1]
3749   {
3750     \tl_gset:Nn \g__enumext_item_symbol_aux_vii_tl {#1}
3751     \__enumext_starred_item_vii_aux_ii:w
```

```
3752        }
3753  \cs_new_protected:Npn \__enumext_starred_item_vii_aux_ii:w
3754    {
3755      \peek_meaning:NTF [
3756        { \__enumext_starred_item_vii_aux_iii:w }
3757        {
3758          \dim_set_eq:NN
3759            \l__enumext_item_symbol_sep_vii_dim
3760            \l__enumext_labelsep_vii_dim
3761          \legacy_if_set_true:n { @noitemarg }
3762          \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
3763        }
3764    }
3765  \cs_new_protected:Npn \__enumext_starred_item_vii_aux_iii:w [#1]
3766    {
3767      \dim_set:Nn \l__enumext_item_symbol_sep_vii_dim {#1}
3768      \legacy_if_set_true:n { @noitemarg }
3769      \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
3770    }
```

(*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.

```
3771  \cs_new_protected_nopar:Npn \__enumext_start_item_vii:w [#1]
3772    {
3773      \cs_set_eq:NN \__enumext_stop_item_tmp_vii: \__enumext_stop_item_vii:
3774      \legacy_if:nT { @noitemarg }
3775        {
3776          \legacy_if_set_false:n { @noitemarg }
3777          \legacy_if:nT { @nmbrlist }
3778            {
3779              \bool_if:NT \l__enumext_hyperref_bool
3780                {
3781                  \legacy_if_set_true:n { @hyper@item }
3782                }
3783              \refstepcounter{enumXvii}
3784              \bool_if:NT \l__enumext_check_answers_bool
3785                {
3786                  \int_gincr:N \g__enumext_item_number_int
3787                }
3788            }
3789        }
```

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.

```
3790        \group_begin:
3791          \lrbox{ \l__enumext_item_text_vii_box }
3792            \bool_if:NF \l__enumext_footnotes_key_bool
3793              {
3794                \__enumext_renew_footnote:
3795              }
3796            \bool_if:NT \l__enumext_item_starred_vii_bool
3797              {
3798                \tl_if_blank:VT \g__enumext_item_symbol_aux_vii_tl
3799                  {
3800                    \tl_gset_eq:NN
3801                      \g__enumext_item_symbol_aux_vii_tl \l__enumext_item_symbol_vii_tl
3802                  }
```

```
3803            \mode_leave_vertical:
3804            \skip_horizontal:n { -\l__enumext_item_symbol_sep_vii_dim }
3805            \makebox[ 0pt ][ r ]{ \g__enumext_item_symbol_aux_vii_tl }
3806            \skip_horizontal:N \l__enumext_item_symbol_sep_vii_dim
3807            \tl_gclear:N \g__enumext_item_symbol_aux_vii_tl
3808          }
3809        \group_begin:
3810          \tl_use:N \l__enumext_label_font_style_vii_tl
3811          \bool_if:NTF \l__enumext_wrap_label_vii_bool
3812            {
3813              \makebox[ \l__enumext_labelwidth_vii_dim ][ \l__enumext_align_label_vii_str ]
3814                { \__enumext_wrapper_label_vii:n {#1} }
3815            }
3816            {
3817              \makebox[ \l__enumext_labelwidth_vii_dim ][ \l__enumext_align_label_vii_str ]{ #1 }
3818            }
3819        \group_end:
3820        \skip_horizontal:N \l__enumext_labelsep_vii_dim
3821        \tl_use:N \l__enumext_after_list_args_vii_tl
3822        \__enumext_minipage:w [ t ]{ \l__enumext_joined_width_vii_dim  }
3823          \skip_set_eq:NN \parindent \l__enumext_listparindent_vii_dim
3824          \skip_set_eq:NN \parskip \l__enumext_parsep_vii_skip
3825          \tl_use:N \l__enumext_fake_item_indent_vii_tl
3826      }
```

(*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.

```
3827 \cs_new_protected_nopar:Nn \__enumext_stop_item_vii:
3828   {
3829         \__enumext_endminipage:
3830       \endlrbox
3831     \group_end:
3832     \box_set_wd:Nn \l__enumext_item_text_vii_box
3833       {
3834         \l__enumext_joined_width_vii_dim
3835         + \l__enumext_labelwidth_vii_dim
3836         + \l__enumext_labelsep_vii_dim
3837       }
3838     \int_set:Nn \hbadness { 10000 }
3839     \box_use:N \l__enumext_item_text_vii_box
3840     \bool_if:NF \l__enumext_footnotes_key_bool
3841       {
3842         \__enumext_print_footnote:
3843       }
3844     \int_compare:nNnTF { \l__enumext_item_column_pos_vii_int } = { \l__enumext_columns_vii_int }
3845       {
3846         \par\noindent
3847         \int_zero:N \l__enumext_item_column_pos_vii_int
3848       }
3849       { \hspace{ \l__enumext_columns_sep_vii_dim } }
3850   }
```

(*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.

```
3851 \cs_new_protected:Nn \__enumext_remove_extra_parsep_vii:
3852   {
3853     \int_compare:nNnT
3854       {
3855         \int_mod:nn {  \g__enumext_item_count_all_vii_int } { \l__enumext_columns_vii_int }
3856       }
3857       =
3858       { \c_zero_int }
3859       {
3860         \par
3861         \vspace{ -\l__enumext_itemsep_vii_skip }
3862         \int_gzero:N \g__enumext_item_count_all_vii_int
```

```
3863          }
3864      }
```

(*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.

```
3865 \__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.

```
3866 \cs_new_protected:Nn \__enumext_starred_columns_set_viii:
3867   {
3868     \dim_compare:nNnT { \l__enumext_columns_sep_viii_dim } = { \c_zero_dim }
3869       {
3870         \dim_set:Nn \l__enumext_columns_sep_viii_dim
3871           {
3872             ( \l__enumext_labelwidth_viii_dim + \l__enumext_labelsep_viii_dim )
3873             / \l__enumext_columns_viii_int
3874           }
3875       }
3876     \int_set:Nn \l__enumext_tmpa_viii_int { \l__enumext_columns_viii_int - \c_one_int }
3877     \dim_set:Nn \l__enumext_item_width_viii_dim
3878       {
3879         ( \linewidth - \l__enumext_columns_sep_viii_dim * \l__enumext_tmpa_viii_int )
3880         / \l__enumext_columns_viii_int - \l__enumext_labelwidth_viii_dim
3881         - \l__enumext_labelsep_viii_dim
3882       }
3883     \dim_zero_new:N \itemwidth
3884   }
```

(*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.

```
3885 \cs_new_protected:Npn \__enumext_starred_joined_item_viii:n #1
3886   {
3887     \int_set:Nn \l__enumext_joined_item_viii_int {#1}
3888     \int_compare:nNnT { \l__enumext_joined_item_viii_int } > { \l__enumext_columns_viii_int }
3889       {
3890         \msg_warning:nnee { enumext } { item-joined }
3891           { \int_use:N \l__enumext_joined_item_viii_int }
3892           { \int_use:N \l__enumext_columns_viii_int }
3893         \int_set:Nn \l__enumext_joined_item_viii_int
3894           {
3895             \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + \c_one_int
3896           }
3897       }
3898     \int_compare:nNnT
3899       { \l__enumext_joined_item_viii_int }
3900       >
3901       { \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + \c_one_int }
3902       {
3903         \msg_warning:nnee { enumext } { item-joined-columns }
3904           { \int_use:N \l__enumext_joined_item_viii_int }
3905           {
3906             \int_eval:n
3907               { \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + \c_one_int }
3908           }
3909         \int_set:Nn \l__enumext_joined_item_viii_int
3910           {
3911             \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + \c_one_int
3912           }
3913       }
3914     \int_compare:nNnTF { \l__enumext_joined_item_viii_int } > { \c_one_int }
3915       {
3916         \int_set_eq:NN \l__enumext_joined_item_aux_viii_int \l__enumext_joined_item_viii_int
3917         \int_decr:N \l__enumext_joined_item_aux_viii_int
```

```
3918          \int_add:Nn \l__enumext_item_column_pos_viii_int { \l__enumext_joined_item_aux_viii_int }
3919          \int_gadd:Nn \g__enumext_item_count_all_viii_int { \l__enumext_joined_item_aux_viii_int }
3920          \dim_set:Nn \l__enumext_joined_width_viii_dim
3921            {
3922              \l__enumext_item_width_viii_dim * \l__enumext_joined_item_viii_int
3923              + (  \l__enumext_labelwidth_viii_dim + \l__enumext_labelsep_viii_dim
3924                + \l__enumext_columns_sep_viii_dim
3925              )*\l__enumext_joined_item_aux_viii_int
3926            }
3927          \dim_set_eq:NN \itemwidth \l__enumext_joined_width_viii_dim
3928        }
3929        {
3930          \dim_set_eq:NN \l__enumext_joined_width_viii_dim \l__enumext_item_width_viii_dim
3931          \dim_set_eq:NN \itemwidth \l__enumext_item_width_viii_dim
3932        }
3933    }
```

(*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:

```
3934  \cs_new_protected:Nn \__enumext_start_mini_viii:
3935    {
3936      \dim_compare:nNnT { \l__enumext_minipage_right_viii_dim } > { \c_zero_dim }
3937        {
3938          \dim_set:Nn \l__enumext_minipage_left_viii_dim
3939            {
3940              \linewidth
3941              - \l__enumext_minipage_right_viii_dim
3942              - \l__enumext_minipage_hsep_viii_dim
3943            }
3944          \bool_set_true:N \l__enumext_minipage_active_viii_bool
3945          \dim_gset_eq:NN
3946            \g__enumext_minipage_right_viii_dim
3947            \l__enumext_minipage_right_viii_dim
3948          \__enumext_mini_addvspace_viii:
3949          \nointerlineskip\noindent
3950          \begin{__enumext_mini_env*}{ \l__enumext_minipage_left_viii_dim }
3951        }
3952    }
3953  \cs_new_protected:Nn \__enumext_stop_mini_viii:
3954    {
3955      \bool_if:NT \l__enumext_minipage_active_viii_bool
3956        {
3957          \end{__enumext_mini_env*}
3958          \hfill
3959          \bool_gset_true:N \g__enumext_minipage_active_viii_bool
3960        }
3961    }
3962  \__enumext_after_env:nn {keyans*}
3963    {
3964      \bool_if:NT \g__enumext_minipage_active_viii_bool
3965        {
3966          \begin{__enumext_mini_env*}{ \g__enumext_minipage_right_viii_dim }
3967            \par\addvspace { \g__enumext_minipage_right_skip }
3968            \bool_if:NF \g__enumext_minipage_center_viii_bool
3969              {
3970                \centering
3971              }
3972            \tl_use:N \g__enumext_miniright_code_viii_tl % the code
3973          \end{__enumext_mini_env*}
3974          \par\addvspace{ \g__enumext_minipage_after_skip }
3975        }
3976      \bool_gset_false:N \g__enumext_minipage_active_viii_bool
3977      \bool_gset_true:N \g__enumext_minipage_center_viii_bool
3978      \tl_gclear:N \g__enumext_miniright_code_viii_tl
3979      \dim_gzero:N \g__enumext_minipage_right_viii_dim
3980    }
```

(*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.

```
3981  \NewDocumentEnvironment{keyans*}{ o }
3982    {
3983      \__enumext_safe_exec_viii:
3984      \__enumext_parse_keys_viii:n {#1}
3985      \__enumext_before_list_viii:
3986      \__enumext_start_list:nn { }
3987        {
3988          \__enumext_list_arg_two_viii:
3989          \__enumext_before_keys_exec_viii:
3990        }
3991      \__enumext_starred_columns_set_viii:
3992      \item[] \scan_stop:
3993      \cs_set_eq:NN \__enumext_stop_item_tmp_viii: \noindent
3994      \cs_set_eq:NN \item \__enumext_start_item_tmp_viii:
3995    }
3996    {
3997      \__enumext_stop_item_tmp_viii:
3998      \__enumext_remove_extra_parsep_viii:
3999      \__enumext_check_starred_cmd:n { item }
4000      \__enumext_stop_list:
4001      \__enumext_after_list_viii:
4002    }
```

(*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.

```
4003  \cs_new_protected:Nn \__enumext_safe_exec_viii:
4004    {
4005      \int_incr:N \l__enumext_keyans_level_h_int
4006      \int_compare:nNnT { \l__enumext_keyans_level_h_int } > { 1 }
4007        {
4008          \msg_error:nn { enumext } { nested }
4009        }
4010      \__enumext_keyans_start_line:
4011      % Set false for interfering with enumext nested in keyans* (yes, its possible and crayze)
4012      \bool_set_false:N \l__enumext_store_active_bool
4013      \int_compare:nNnT { \l__enumext_level_int } > { 1 }
4014        {
4015          \msg_error:nn { enumext } { keyans-wrong-level }
4016        }
4017    }
```

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

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

```
4018  \cs_new_protected:Npn \__enumext_parse_keys_viii:n #1
4019    {
4020      \tl_if_novalue:nF {#1}
4021        {
4022          \keys_set:nn { enumext / keyans* } {#1}
4023        }
4024    }
```

(*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`.

```
4025  \cs_new_protected:Nn \__enumext_before_list_viii:
4026    {
4027      \__enumext_vspace_above_viii:
4028      \__enumext_before_args_exec_viii:
4029      \__enumext_start_mini_viii:
4030    }
```

(*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.

```
4031 \cs_new_protected:Nn \__enumext_after_list_viii:
4032   {
4033     \__enumext_stop_mini_viii:
4034     \__enumext_after_stop_list_viii:
4035     \__enumext_vspace_below_viii:
4036   }
```

(*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.

```
4037 \cs_new_protected_nopar:Nn \__enumext_start_item_tmp_viii:
4038   {
4039     \__enumext_stop_item_tmp_viii:
4040     \int_incr:N \l__enumext_item_column_pos_viii_int
4041     \int_gincr:N \g__enumext_item_count_all_viii_int
4042     \__enumext_item_peek_args_viii:
4043   }
```

(*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).

```
4044 \cs_new_protected:Nn \__enumext_item_peek_args_viii:
4045   {
4046     \peek_meaning:NTF (
4047       { \__enumext_joined_item_viii:w }
4048       { \__enumext_joined_item_viii:w (1) }
4049   }
```

(*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.

```
4050 \cs_new_protected:Npn \__enumext_joined_item_viii:w (#1)
4051   {
4052     \__enumext_starred_joined_item_viii:n {#1}
4053     \peek_meaning_remove:NTF *
4054       { \__enumext_starred_item_viii:w  }
4055       { \__enumext_standar_item_viii:w }
4056   }
```

(*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].

```
4057 \cs_new_protected:Npn \__enumext_standar_item_viii:w
4058   {
4059     \bool_set_false:N \l__enumext_item_starred_viii_bool
4060       \peek_meaning:NTF [
4061         {
4062           \bool_set_eq:NN
4063             \l__enumext_wrap_label_viii_bool
4064             \l__enumext_wrap_label_opt_viii_bool
4065           \__enumext_start_item_viii:w
4066         }
4067         {
4068           \bool_set_true:N \l__enumext_wrap_label_viii_bool
4069           \legacy_if_set_true:n { @noitemarg }
4070           \__enumext_start_item_viii:w [ \l__enumext_label_viii_tl ]
4071         }
4072   }
```

(*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*⟩].

```
4073 \cs_new_protected:Npn \__enumext_starred_item_viii:w
4074   {
4075     \bool_set_true:N \l__enumext_item_starred_viii_bool
4076     \bool_set_true:N \l__enumext_wrap_label_viii_bool
4077     \peek_meaning:NTF [
4078       { \__enumext_starred_item_viii_aux_i:w }
4079       { \__enumext_starred_item_viii_aux_ii:w }
4080   }
```

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.

```
4081 \cs_new_protected:Npn \__enumext_starred_item_viii_aux_i:w [#1]
4082   {
4083     \tl_clear:N \l__enumext_store_current_label_tl
4084     \tl_if_novalue:nF { #1 }
4085       {
4086         \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_tl
4087           {
4088             \tl_put_right:Ne \l__enumext_store_current_label_tl { \l__enumext_store_keyans_item_o|
4089             \tl_put_right:Ne \l__enumext_store_current_label_tl { #1 }
4090           }
4091         \tl_set:Ne \l__enumext_store_current_opt_arg_tl { #1 }
4092       }
4093     \__enumext_starred_item_viii_aux_ii:w
4094   }
4095 \cs_new_protected:Npn \__enumext_starred_item_viii_aux_ii:w
4096   {
4097     \legacy_if_set_true:n { @noitemarg }
4098     \__enumext_start_item_viii:w [ \l__enumext_label_viii_tl ]
4099   }
```

(*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.

```
4100 \cs_new_protected:Nn \__enumext_starred_item_exec:
4101   {
4102     \tl_put_left:Ne \l__enumext_store_current_label_tl { \l__enumext_label_viii_tl }
4103     \__enumext_store_addto_prop:V \l__enumext_store_current_label_tl
4104     \__enumext_keyans_store_ref:
4105     \tl_put_left:Ne \l__enumext_store_current_label_tl { \item }
4106     \__enumext_keyans_addto_seq_link:
4107     \int_gincr:N \g__enumext_check_starred_cmd_int
4108     \bool_if:NT \l__enumext_show_answer_bool
4109       {
```

```
4110            \__enumext_print_keyans_box:NN \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
4111          }
4112        \bool_if:NT \l__enumext_show_position_bool
4113          {
4114            \tl_set:Ne \l__enumext_mark_answer_sym_tl
4115              {
4116                \group_begin:
4117                  \exp_not:N \normalfont
4118                  \exp_not:N \footnotesize [ \int_eval:n
4119                    {
4120                      \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
4121                    }
4122                  ]
4123                \group_end:
4124              }
4125            \__enumext_print_keyans_box:NN \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
4126          }
4127      }
```

(*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.

```
4128 \cs_new_protected_nopar:Npn \__enumext_start_item_viii:w [#1]
4129   {
4130      \cs_set_eq:NN \__enumext_stop_item_tmp_viii: \__enumext_stop_item_viii:
4131      \legacy_if:nT { @noitemarg }
4132        {
4133          \legacy_if_set_false:n { @noitemarg }
4134          \legacy_if:nT { @nmbrlist }
4135            {
4136              \bool_if:NT \l__enumext_hyperref_bool
4137                {
4138                  \legacy_if_set_true:n { @hyper@item }
4139                }
4140              \refstepcounter{enumXviii}
4141            }
4142        }
```

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

```
4143        \group_begin:
4144          \lrbox{ \l__enumext_item_text_viii_box }
4145            \bool_if:NF \l__enumext_footnotes_key_bool
4146              {
4147                \__enumext_renew_footnote:
4148              }
4149            \bool_if:NT \l__enumext_item_starred_viii_bool
4150              {
4151                \__enumext_starred_item_exec:
4152              }
4153            \group_begin:
4154              \tl_use:N \l__enumext_label_font_style_viii_tl
4155              \bool_if:NTF \l__enumext_wrap_label_viii_bool
4156                {
4157                  \makebox[ \l__enumext_labelwidth_viii_dim ][ \l__enumext_align_label_viii_str ]
4158                    { \__enumext_wrapper_label_viii:n {#1} }
4159                }
4160                {
4161                  \makebox[ \l__enumext_labelwidth_viii_dim ][ \l__enumext_align_label_viii_str ]{ #1
4162                }
4163            \group_end:
4164            \skip_horizontal:N \l__enumext_labelsep_viii_dim
4165            \tl_use:N \l__enumext_after_list_args_viii_tl
4166            \__enumext_minipage:w [ t ]{ \l__enumext_joined_width_viii_dim  }
4167              \skip_set_eq:NN \parindent \l__enumext_listparindent_viii_dim
4168              \skip_set_eq:NN \parskip \l__enumext_parsep_viii_skip
4169              \bool_if:NT \l__enumext_item_starred_viii_bool
4170                {
4171                  \tl_use:N \l__enumext_fake_item_indent_viii_tl
4172                  \__enumext_keyans_show_item_opt:
4173                  \skip_horizontal:n { -\l__enumext_fake_item_indent_viii_dim - \l__enumext_labelsep_
```

```
4174                    }
4175                    {
4176                        \tl_use:N \l__enumext_fake_item_indent_viii_tl
4177                    }
4178            }
```

(*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.

```
4179  \cs_new_protected_nopar:Nn \__enumext_stop_item_viii:
4180    {
4181            \__enumext_endminipage:
4182        \endlrbox
4183      \group_end:
4184      \box_set_wd:Nn \l__enumext_item_text_viii_box
4185        {
4186          \l__enumext_joined_width_viii_dim
4187          + \l__enumext_labelwidth_viii_dim
4188          + \l__enumext_labelsep_viii_dim
4189        }
4190      \int_set:Nn \hbadness { 10000 }
4191      \box_use:N \l__enumext_item_text_viii_box
4192      \bool_if:NF \l__enumext_footnotes_key_bool
4193        {
4194          \__enumext_print_footnote:
4195        }
4196      \int_compare:nNnTF
4197        { \l__enumext_item_column_pos_viii_int } = { \l__enumext_columns_viii_int }
4198        {
4199          \par\noindent
4200          \int_zero:N \l__enumext_item_column_pos_viii_int
4201        }
4202        { \hspace{ \l__enumext_columns_sep_viii_dim } }
4203    }
```

(*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.

```
4204  \cs_new_protected:Nn \__enumext_remove_extra_parsep_viii:
4205    {
4206      \int_compare:nNnT
4207        {
4208          \int_mod:nn
4209            { \g__enumext_item_count_all_viii_int }
4210            { \l__enumext_columns_viii_int }
4211        }
4212        =
4213        { \c_zero_int }
4214        {
4215          \par
4216          \vspace{ -\l__enumext_itemsep_viii_skip }
4217          \int_gzero:N \g__enumext_item_count_all_viii_int
4218        }
4219    }
```

(*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.

```
4220  \NewDocumentCommand \getkeyans { m }
4221    {
4222      \exp_args:Ne \__enumext_getkeyans_aux:n
4223        { \tl_to_str:e { \text_expand:n {#1} } }
4224    }
```

(*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.

```
4225 \cs_new_protected:Npn \__enumext_getkeyans_aux:n #1
4226   {
4227     \str_if_in:nnTF {#1} { : }
4228       {
4229         \use:e
4230           {
4231             \cs_set:Npn \exp_not:N \__enumext_tmp:w ##1 \c_colon_str ##2 \scan_stop:
4232               { {##1} {##2} }
4233           }
4234         \exp_after:wN \__enumext_getkeyans:nn \__enumext_tmp:w #1 \scan_stop:
4235       }
4236     { \msg_error:nnn { enumext } { missing-colon } {#1} }
4237   }
```

(*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*⟩.

```
4238 \cs_new_protected:Npn \__enumext_getkeyans:nn #1 #2
4239   {
4240     \prop_if_exist:cF { g__enumext_#1_prop }
4241       { \msg_error:nnn { enumext } { undefined-storage-anskey } {#1} }
4242     \group_begin:
4243       \prop_item:cn { g__enumext_#1_prop }{#2}
4244     \group_end:
4245   }
```

(*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*⟩].

```
4246 \cs_generate_variant:Nn \keys_precompile:nnN { neN }
4247 \keys_define:nn  { enumext / print }
4248   {
4249     print*  .code:n     = \keys_precompile:neN { enumext / enumext* }
4250                            { \__enumext_filter_save_key:n {#1} }
4251                          \l__enumext_print_keyans_starred_tl, % starred cmd
4252     print*  .initial:n = { nosep, label=\arabic*., columns=2, first=\small, font=\small },
4253     print-1 .code:n     = \keys_precompile:neN { enumext / level-1 }
4254                            { \__enumext_filter_save_key:n {#1} }
4255                          \l__enumext_print_keyans_i_tl,
4256     print-1 .initial:n = { nosep, label=\arabic*., columns=2, first=\small, font=\small },
4257     print-2 .code:n     = \keys_precompile:neN { enumext / level-2 }
4258                            { \__enumext_filter_save_key:n {#1} }
4259                          \l__enumext_print_keyans_ii_tl,
4260     print-2 .initial:n = { nosep, label=(\alph*), first=\small, font=\small },
4261     print-3 .code:n     = \keys_precompile:neN { enumext / level-3 }
4262                            { \__enumext_filter_save_key:n {#1} }
4263                          \l__enumext_print_keyans_iii_tl,
4264     print-3 .initial:n = { nosep, label=\roman*., first=\small, font=\small },
4265     print-4 .code:n     = \keys_precompile:neN { enumext / level-4 }
4266                            { \__enumext_filter_save_key:n {#1} }
4267                          \l__enumext_print_keyans_iv_tl,
4268     print-4 .initial:n = { nosep, label=\Alph*., first=\small, font=\small },
4269     print-* .code:n     = \keys_precompile:neN { enumext / enumext* }
4270                            { \__enumext_filter_save_key:n {#1} }
4271                          \l__enumext_print_keyans_vii_tl, % starred nested
4272     print-* .initial:n = { nosep, label=\arabic*., first=\small, font=\small },
4273   }
```

💣 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.

```
4274 \NewDocumentCommand \printkeyans { s O{} m }
4275   {
4276     \group_begin:
4277       \tl_use:N \l__enumext_print_keyans_i_tl
4278       \tl_use:N \l__enumext_print_keyans_ii_tl
4279       \tl_use:N \l__enumext_print_keyans_iii_tl
4280       \tl_use:N \l__enumext_print_keyans_iv_tl
4281       \tl_use:N \l__enumext_print_keyans_vii_tl
4282       \__enumext_printkeyans:nnn { #1 } { #2 } { #3 }
4283     \group_end:
4284   }
```

(*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.

```
4285 \cs_new_protected:Npn \__enumext_printkeyans:nnn #1 #2 #3
4286   {
4287     \seq_if_exist:cTF { g__enumext_#3_seq }
4288       {
4289         \seq_if_empty:cF { g__enumext_#3_seq }
4290           {
4291             %%\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⟩

```
4292             \bool_if:nTF {#1}
4293               {
4294                 \seq_if_in:cnTF { g__enumext_#3_seq } { \end{enumext*} }
4295                   {
4296                     \msg_error:nnnn { enumext } { print-starred } {#3} { enumext* }
4297                   }
4298                   {
4299                     \tl_use:N \l__enumext_print_keyans_starred_tl
4300                     \begin{enumext*}[#2]
4301                       \seq_map_inline:cn { g__enumext_#3_seq } { ##1 }
4302                     \end{enumext*}
4303                   }
4304               }
```

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

```
4305               {
4306                 \begin{enumext}[#2]
4307                   \seq_map_inline:cn { g__enumext_#3_seq } { ##1 }
4308                 \end{enumext}
4309               }
4310           }
4311       }
4312       {
4313         \msg_error:nnn { enumext } { undefined-storage-anskey } {#3}
4314       }
4315   }
```

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

11.41 The command \setenumext

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

```
4316  \keys_define:nn { enumext / meta-families }
4317    {
4318      enumext-1 .code:n = { \keys_set:nn { enumext / level-1  } {#1} } ,
4319      enumext-2 .code:n = { \keys_set:nn { enumext / level-2  } {#1} } ,
4320      enumext-3 .code:n = { \keys_set:nn { enumext / level-3  } {#1} } ,
4321      enumext-4 .code:n = { \keys_set:nn { enumext / level-4  } {#1} } ,
4322      keyans    .code:n = { \keys_set:nn { enumext / keyans   } {#1} } ,
4323      enumext*  .code:n = { \keys_set:nn { enumext / enumext*  } {#1} } ,
4324      keyans*   .code:n = { \keys_set:nn { enumext / keyans*  } {#1} } ,
4325      print*    .code:n = { \keys_set:nn { enumext / print     } { print*  = {#1} } } ,
4326      print-1   .code:n = { \keys_set:nn { enumext / print     } { print-1 = {#1} } } ,
4327      print-2   .code:n = { \keys_set:nn { enumext / print     } { print-2 = {#1} } } ,
4328      print-3   .code:n = { \keys_set:nn { enumext / print     } { print-3 = {#1} } } ,
4329      print-4   .code:n = { \keys_set:nn { enumext / print     } { print-4 = {#1} } } ,
4330      print-*   .code:n = { \keys_set:nn { enumext / print     } { print-* = {#1} } } ,
4331      unknown   .code:n = { \msg_error:nn { enumext } { unknown-key-family } } ,
4332    }
```

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

```
4333  \seq_const_from_clist:Nn \c__enumext_all_families_seq
4334    {
4335      enumext-1, enumext-2, enumext-3, enumext-4, keyans, enumext*,
4336      keyans*, print-1, print-2, print-3, print-4, print-*, print*,
4337    }
```

\setenumext Now we define the user command \setenumext.

```
4338  \NewDocumentCommand \setenumext { O{enumext,1} +m }
4339    {
4340      \tl_if_novalue:nTF {#1}
4341        {
4342          \seq_map_inline:Nn \c__enumext_all_families_seq
4343        }
4344        {
4345          \seq_clear:N \l__enumext_setkey_tmpa_seq
4346          \seq_set_from_clist:Nn \l__enumext_setkey_tmpb_seq {#1}
4347          \int_set:Nn \l__enumext_setkey_tmpa_int
4348            {
4349              \seq_count:N \l__enumext_setkey_tmpb_seq
4350            }
4351          \int_compare:nNnTF { \l__enumext_setkey_tmpa_int } > { 1 }
4352            {
4353              \seq_pop_left:NN \l__enumext_setkey_tmpb_seq \l__enumext_setkey_tmpa_tl
4354              \seq_map_function:NN \l__enumext_setkey_tmpb_seq \__enumext_set_parse:n
4355              \seq_set_map_e:NNn \l__enumext_setkey_tmpa_seq \l__enumext_setkey_tmpa_seq
4356                {
4357                  \tl_use:N \l__enumext_setkey_tmpa_tl - ##1
4358                }
4359            }
4360            {
4361              \seq_put_right:Ne \l__enumext_setkey_tmpa_seq { \tl_trim_spaces:n {#1} }
4362            }
4363          \seq_if_empty:NTF \l__enumext_setkey_tmpa_seq
4364            { \seq_map_inline:Nn \c__enumext_all_families_seq }
4365            { \seq_map_inline:Nn \l__enumext_setkey_tmpa_seq }
4366        }
4367        {
4368          \keys_set:nn { enumext / meta-families } { ##1 = {#2} }
4369        }
4370    }
```

(*End of definition for* \setenumext. *This function is documented on page* 6.)

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

```
4371  \cs_new_protected:Npn \__enumext_set_parse:n #1
4372    {
4373      \tl_set:Ne \l__enumext_setkey_tmpb_tl { \tl_trim_spaces:n {#1} }
4374      \clist_map_inline:nn { 0, 1, 2, 3, 4, * } % <- max level
4375        { \tl_remove_all:Nn \l__enumext_setkey_tmpb_tl {##1} }
```

```
4376      \tl_if_empty:NTF \l__enumext_setkey_tmpb_tl
4377        {
4378          \seq_put_right:Ne \l__enumext_setkey_tmpa_seq
4379            { \tl_trim_spaces:n {#1} }
4380        }
4381        { \__enumext_set_error:nn {#1} { } }
4382    }
4383 \cs_new_protected:Npn \__enumext_set_error:nn #1 #2
4384    { \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.

```
4385 \msg_new:nnn { enumext } { package-load }
4386    {
4387      The ~ '#1' ~ package ~ is ~ already ~ loaded.
4388    }
4389 \msg_new:nnn { enumext } { package-not-load }
4390    {
4391      The ~ '#1' ~ package ~ will ~ be ~ loaded ~ as ~ a ~ dependency.
4392    }
4393 \msg_new:nnn { enumext } { package-load-foot }
4394    {
4395      The ~ '#1' ~ package ~ is ~ loaded ~ with ~ the ~ option ~ '#2'.
4396    }
```

Message used in the creation of counters by enumext package.

```
4397 \msg_new:nnn { enumext } { counters }
4398    {
4399      The ~ counter ~ '#1' ~ is ~ already ~ defined ~ by ~ some ~ \\
4400      package ~ or ~ macro, ~ it ~ cannot ~ be ~ continued.
4401    }
```

Message used by align and mark-pos keys.

```
4402 \msg_new:nnn { enumext } { unknown-choice }
4403    {
4404      The ~ value ~ '#3' ~ for ~ '#1' ~ key ~ is ~ invalid ~ use ~ ('#2').
4405    }
4406 %    \begin{macrocode}
4407 %
4408 % Message used by reserved \myenv*{anskey*} environment by \mypkg*{enumext} package.
4409 %    \begin{macrocode}
4410 \msg_new:nnnn { enumext } { anskey-env-error }
4411    {
4412      The ~ '#1' ~ environment ~is ~ reserved ~ by ~\\
4413      'enumext' ~ package, ~ It~ is~ already~ defined.
4414    }
4415    {
4416      The ~ anskey* ~ environment ~ is ~ defined ~ internally ~
4417      for ~ the ~ 'save-ans' ~ key.\\
4418    }
```

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

```
4419 \msg_new:nnn { enumext } { store-prop }
4420    {
4421      * ~ Package ~ enumext: ~ Creating ~
4422      \c_backslash_str g__enumext_#1_prop ~ \msg_line_context:.
4423    }
4424 \msg_new:nnn { enumext } { store-seq }
4425    {
4426      * ~ Package ~ enumext: ~ Creating ~
4427      \c_backslash_str g__enumext_#1_seq ~ \msg_line_context:.
4428    }
4429 \msg_new:nnn { enumext } { store-int }
4430    {
4431      * ~ Package ~ enumext: ~ Creating ~
4432      \c_backslash_str g__enumext_resume_#1_int ~ \msg_line_context:.
4433    }
4434 \msg_new:nnn { enumext } { prop-seq-int-hook }
4435    {
4436      * ~ Package ~ enumext: ~ Elements ~ in ~
```

```
4437        \c_backslash_str g__enumext_#1_prop ~ = ~ #2.\\
4438        * ~ Package ~ enumext: ~ Elements ~ in ~
4439        \c_backslash_str g__enumext_#1_seq ~ = ~ #3.\\
4440        * ~ Package ~ enumext: ~ Value ~ off ~
4441        \c_backslash_str g__enumext_resume_#1_int ~ = ~ #4.
4442      }
4443    \msg_new:nnn { enumext } { item-answer-hook }
4444      {
4445        * ~ Package ~ enumext: ~ Value ~ off ~
4446        \c_backslash_str g__enumext_item_number_int ~ = ~ #1.\\
4447        * ~ Package ~ enumext: ~ Value ~ off ~
4448        \c_backslash_str g__enumext_item_anskey_int ~ = ~ #2.\\
4449        * ~ Package ~ enumext: ~ Difference ~ item_number_int ~ - ~ item_anskey_int ~ = ~ #3.
4450      }
```

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

```
4451    \msg_new:nnn { enumext } { invalid-key }
4452      {
4453        The ~ key ~ '#1' ~ is ~ not ~ know ~ the ~ level ~ #2.
4454      }
4455    \msg_new:nnn { enumext } { unknown-key-family }
4456      {
4457        Unknown~key~family~`\l_keys_key_str'~for~enumext.
4458      }
```

Messages used in length calculation.

```
4459    \msg_new:nnn { enumext } { width-negative }
4460      {
4461        Ignoring ~ negative ~ value ~ '#1=#2' ~ \msg_line_context:.\\
4462        The ~ key ~ '#1'~ accepts ~ values  ~ >= ~ 0pt.
4463      }
4464    \msg_new:nnn { enumext } { width-zero }
4465      {
4466        Invalid ~ '#1=#2' ~ \msg_line_context:.\\
4467        The ~ key ~ '#1'~ accepts ~ values  ~ > ~ 0pt.
4468      }
```

Messages used by show-length key in enumext.

```
4469    \msg_new:nnn { enumext } { list-lengths }
4470      {
4471        **** ~ Lengths ~ used ~ by ~ 'enumext' ~ level ~ '#2' ~ \msg_line_context:~\c_space_tl ****\\
4472        \__enumext_show_length:nnn { dim  } { labelsep      } {#1}
4473        \__enumext_show_length:nnn { dim  } { labelwidth    } {#1}
4474        \__enumext_show_length:nnn { dim  } { itemindent    } {#1}
4475        \__enumext_show_length:nnn { dim  } { leftmargin    } {#1}
4476        \__enumext_show_length:nnn { dim  } { rightmargin   } {#1}
4477        \__enumext_show_length:nnn { dim  } { listparindent } {#1}
4478        \__enumext_show_length:nnn { skip } { topsep     } {#1}
4479        \__enumext_show_length:nnn { skip } { parsep     } {#1}
4480        \__enumext_show_length:nnn { skip } { partopsep  } {#1}
4481        \__enumext_show_length:nnn { skip } { itemsep    } {#1}
4482        *************************************************
4483      }
```

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

```
4484    \msg_new:nnn { enumext } { list-lengths-not-nested }
4485      {
4486        **** ~ Lengths ~ used ~ by ~ '#2' ~ environment ~ \msg_line_context:~\c_space_tl ****\\
4487        \__enumext_show_length:nnn { dim  } { labelsep      } {#1}
4488        \__enumext_show_length:nnn { dim  } { labelwidth    } {#1}
4489        \__enumext_show_length:nnn { dim  } { itemindent    } {#1}
4490        \__enumext_show_length:nnn { dim  } { leftmargin    } {#1}
4491        \__enumext_show_length:nnn { dim  } { rightmargin   } {#1}
4492        \__enumext_show_length:nnn { dim  } { listparindent } {#1}
4493        \__enumext_show_length:nnn { skip } { topsep     } {#1}
4494        \__enumext_show_length:nnn { skip } { parsep     } {#1}
4495        \__enumext_show_length:nnn { skip } { partopsep  } {#1}
4496        \__enumext_show_length:nnn { skip } { itemsep    } {#1}
4497        *************************************************
4498      }
```

Messages used by ref key.

```
4499    \msg_new:nnn { enumext } { key-ref-empty }
```

```
4500        {
4501          Key ~ 'ref' ~ need ~ a ~ value ~ in ~ '#1'~ \msg_line_context:.
4502        }
```

Messages used by save-ans key.

```
4503  \msg_new:nnn { enumext } { save-ans-empty }
4504        {
4505          Key ~ 'save-ans' ~ need ~ a ~ value ~ in ~ '#1'~ \msg_line_context:.
4506        }
4507  \msg_new:nnn { enumext } { save-ans-log }
4508        {
4509          * ~ Package ~ enumext: ~ Start ~ #1\c_space_tl with ~ save-ans=#2 ~ \msg_line_context:.
4510        }
4511  \msg_new:nnn { enumext } { save-ans-log-hook }
4512        {
4513          * ~ Package ~ enumext: ~ Stop ~ #1\c_space_tl with ~ save-ans=#2 ~ \msg_line_context:.
4514        }
4515  \msg_new:nnn { enumext } { save-ans-hook }
4516        {
4517          Stop ~ storing ~ for ~ 'save-ans=#1' ~ \msg_line_context:.
4518        }
```

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

```
4519  \msg_new:nnn { enumext } { need-save-ans }
4520        {
4521          Key ~ '#1'~ works ~ only ~ with ~ the ~ 'save-ans' ~ key ~ in ~ '#2'~ \msg_line_context:.
4522        }
4523  \msg_new:nnn { enumext } { items-same-answer }
4524        {
4525          ***************************************\\
4526          * ~ Package ~ enumext: ~ Checking ~ answers ~ in ~ '#1' ~
4527          for ~ \c_left_brace_str #2 \c_right_brace_str\\
4528          * ~ started ~ #3 ~ and ~ close ~ \msg_line_context: : ~
4529          'OK', ~ all ~ items ~ with ~ answer.\\
4530          ***************************************
4531        }
4532  \msg_new:nnn { enumext } { item-greater-answer }
4533        {
4534          Checking ~ answers ~ in ~ '#1' ~ for ~ \c_left_brace_str #2 \c_right_brace_str\\
4535          started ~ #3 ~ and ~ close ~ \msg_line_context: : ~'NOT ~ OK'\\
4536          Items ~ > ~ Answers.
4537        }
4538  \msg_new:nnn { enumext } { item-less-answer }
4539        {
4540          Checking ~ answers ~ in ~ '#1' ~ for ~ \c_left_brace_str #2 \c_right_brace_str\\
4541          started ~ #3 ~ and ~ close ~ \msg_line_context: : ~'NOT ~ OK'\\
4542          Items ~ < ~ Answers.
4543        }
```

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

```
4544  \msg_new:nnn { enumext } { missing-starred }
4545        {
4546          Missing ~ '\c_backslash_str #1*' ~ #2.
4547        }
4548  \msg_new:nnn { enumext } { many-starred }
4549        {
4550          Many ~ '\c_backslash_str #1*' ~ #2.
4551        }
```

Messages used by \printkeyans* command.

```
4552  \msg_new:nnn { enumext } { print-starred }
4553        {
4554          \c_backslash_str printkeyans*:~ The ~ sequence ~ '#1' ~ already ~ contains ~
4555          #2 ~ environment ~  \msg_line_context:.
4556        }
```

Message for the nesting depth of the environment enumext.

```
4557  \msg_new:nnn { enumext } { list-too-deep }
4558        {
4559          Too ~ deep ~ nesting  ~ for ~ 'enumext' ~ \msg_line_context:.~ \\
4560          The ~ maximum  ~ level  ~ of  ~ nesting  ~ is ~ 4.
4561        }
```

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

```
4562 \msg_new:nnn { enumext } { anskey-empty-arg }
4563   {
4564     Can't ~ store ~ empty ~ content ~ ~ \msg_line_context:.
4565   }
4566 \msg_new:nnn { enumext } { anskey-wrong-place }
4567   {
4568     Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:.~ \\
4569     '\c_backslash_str #1' ~ works ~ in ~ the ~ environment ~ '#2'.
4570   }
4571 \msg_new:nnn { enumext } { anskey-nested }
4572   {
4573     The ~ command ~ \c_backslash_str anskey~ can't ~ be ~ nested ~ \msg_line_context:.
4574   }
4575 \msg_new:nnn { enumext } { anskey-env-wrong }
4576   {
4577     The ~ environment ~ anskey* ~ cannot ~ use ~ in ~ '#1' ~ \msg_line_context:.
4578   }
4579 \msg_new:nnn { enumext } { anspic-wrong-place }
4580   {
4581     Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:.~ \\
4582     '\c_backslash_str #1' ~ works ~ in ~ the ~ environment ~ '#2'.
4583   }
4584 \msg_new:nnn { enumext } { command-wrong-place }
4585   {
4586     Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:.~ \\
4587     '\c_backslash_str #1' ~ works ~ outside ~ the ~ environment ~ '#2'.
4588   }
```

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

```
4589 \msg_new:nnn { enumext } { keyans-nested }
4590   {
4591     The ~ environment ~ 'keyans' ~ can't ~ be  ~ nested  ~ \msg_line_context:.
4592   }
4593 \msg_new:nnn { enumext } { keyans-wrong-level }
4594   {
4595     Wrong ~ level ~ position ~ for ~ 'keyans' ~ \msg_line_context:.~ \\
4596     The ~ environment ~ 'keyans' ~ can ~ only ~ be ~ in ~ the ~ first ~ level.
4597   }
4598 \msg_new:nnn { enumext } { wrong-place }
4599   {
4600     Wrong ~ place ~ for ~ '#1' ~ environment ~\msg_line_context:.~ \\
4601     '#1' ~ is ~ only ~ found ~ with ~ '#2' ~  in  ~  'enumext.
4602   }
4603 \msg_new:nnn { enumext } { keyanspic-nested }
4604   {
4605     The ~ environment ~ 'keyanspic' ~ can't ~ be  ~ nested~ \msg_line_context:.~.
4606   }
4607 \msg_new:nnn { enumext } { keyanspic-wrong-level }
4608   {
4609     Wrong ~ level ~ position ~ for ~ 'keyanspic' ~ \msg_line_context:.~ \\
4610     The ~ environment ~ 'keyans' ~ can ~ only ~ be ~ in ~ the ~ first ~ level.
4611   }
```

Messages used by `\getkeyans` command.

```
4612 \msg_new:nnn { enumext } { undefined-storage-anskey }
4613   {
4614     Storage ~ named ~ '#1' ~ is ~ not ~ defined ~ \msg_line_context:.
4615   }
```

Messages used by `\miniright` command.

```
4616 \msg_new:nnn { enumext } { missing-miniright }
4617   {
4618     Missing ~ '\c_backslash_str miniright' ~ in ~ \msg_line_context:.\\
4619     The ~ key ~ 'mini-env' ~ need ~ '\c_backslash_str miniright'.
4620   }
4621 \msg_new:nnn { enumext } { wrong-miniright-place }
4622   {
4623     Wrong ~ place ~ for ~ '\c_backslash_str miniright' ~ \msg_line_context:.~ \\
4624     Works ~ in ~ 'enumext' ~ and ~ 'keyans' ~ with ~ key ~ 'mini-env'.
4625   }
4626 \msg_new:nnn { enumext } { wrong-miniright-use }
```

```
4627    {
4628      Wrong ~ use ~ for ~ '\c_backslash_str miniright' ~ \msg_line_context:.~ \\
4629      '\c_backslash_str miniright' ~ need ~ a ~ key ~ 'mini-env'.
4630    }
```

Messages used by enumext* and keyans* environments.

```
4631 \msg_new:nnn { enumext } { nested }
4632    {
4633      The ~ starred ~ environment ~ can't ~ be ~ nested ~ \msg_line_context:.
4634    }
4635 \msg_new:nnn { enumext } { item-joined }
4636    {
4637      Items ~ joined ~ (#1) ~ > ~ #2  ~ columns ~\msg_line_context:.
4638    }
4639 \msg_new:nnn { enumext } { item-joined-columns }
4640    {
4641      Not ~ space ~ to ~ join ~ items ~ (#1) ~ > ~ #2 ~\msg_line_context:.
4642    }
```

11.43   Finish package

Finish package implementation.

```
4643 \file_input_stop:
4644 ⟨/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.