enumext

ENUMERATE EXERCISE SHEETS

V1.0 2024-09-25[*]

©2024 by Pablo González[†]

Abstract

This package provides *"enumerated list"* environments compatible with tagging PDF 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 2024-09-25.
[†]E-mail: «pablgonz@educarchile.cl».

License and Requirements

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

The enumext package loads and requires `multicol`[3] and `scontents`[4] packages, need to have a modern TeX distribution such as TeX Live or MiKTeX. It has been tested with the standard classes provided by LaTeX: `book`, `report`, `article` and `letter` on `10pt`, `11pt` and `12pt`.

1   Introduction

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

4. Question with image and label below:
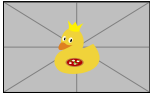

A)   B)   C)

D)   E)

5. Question with image on left side:

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

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

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

* 4. E), A duck
* 5. D), "other note"

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

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

1.1 Description and usage

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

🎯 This package is NOT intend to replace the `enumerate` environment nor replace the powerful `enumitem`[6], the approach is intended to work without hindering either of them.

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

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

The package is loaded in the usual way:

```
\usepackage{enumext}
```

1.2 The concept of left margin

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

Usually we don't want the `list` to go beyond the left margin of the page, but since these four values are related, that causes a problem. The `enumitem`[6] package adds the `\labelindent` parameter to solve some of these problems. A simplified representation of this in the figure 1.

Figure 1: Representation of horizontal lengths in `enumitem`.

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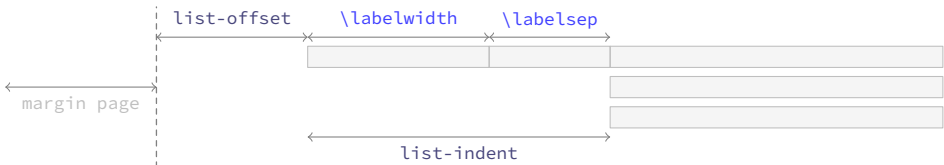
Figure 2: Representation of horizontal lengths concept in enumext.

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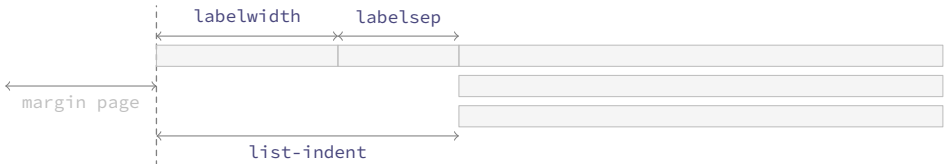

Figure 3: Default horizontal lengths `list-offset=0pt`, `list-indent=\labelwidth+\labelsep` in enumext.

1.3 User interface

The user interface consists of two main list environments `enumext` (vertical) and `enumext*` (horizontal), the environment `anskey*` and the command `\anskey` to *"store content"* and the environments `keyans`, `keyans*` and `keyanspic` for multiple choice. It also provides the commands `\getkeyans` to print individual *stored content*, `\printkeyans` to print all *stored content*, `\miniright` for `minipage` and `\setenumext` to config all [⟨*key = val*⟩] options.

1.3.1 Internal counters

The package enum**ext** 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 Public dimension

The package enum**ext** only provides a single public dimension `\itemwidth` and is intended for user convenience only and is not for internal use as such. The dimension `\itemwidth` is *rigid length* and contains the *"width of the content"* of each `\item` regardless of `labelwidth` and `labelsep`.

🔘 If any package defines `\itemwidth` or they are user-defined `\itemwidth` in the document, the package will overwrite it without warning.

1.3.3 Support for multicol

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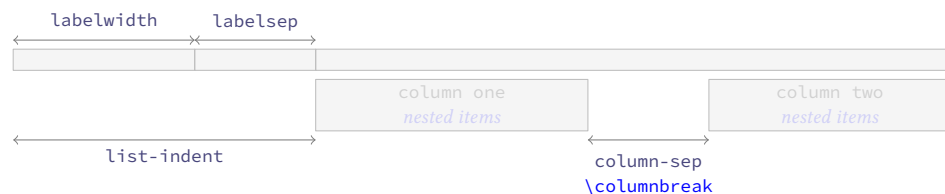

Figure 4: Representation of the two column output for a nested level in `enumext` environment.

The *"non starred"* version of the `multicols` environment is always used together with the `\raggedcolumns` command and is controlled by `columns` and `columns-sep` keys. It can be used in all nesting levels of the environment `enumext` and the environment `keyans` and can together with the `mini-env` key. If you need to force a start a new column `\columnbreak` must be used (see §5.5).

🔘 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.

1.3.4 Support for `minipage`

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

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

The `minipage` environments on *"left side"* and *"right side"* is always used with *"aligned on top"* [t]. It can be used in all nesting levels of the environment `enumext` and the environment `keyans` and is controlled by `mini-env` and `mini-sep` keys. In order to switch from the *"left"* side `minipage` environment to the *"right"* side one must use the command `\miniright` (see §5.6).

1.3.5 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 (§5.1), 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` (§6.1.1) when the key `save-ans` (§6.1) is active.

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

1.3.6 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*}
```

At the moment the footnotehyper package is not compatible with LaTeX *tagged* PDF.

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* 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 or in the environment keyans*, but it can be nested within enumext and vice versa.

- Each *"item"* in the environment is placed within a minipage environment whose *width* is stored in the dimension \itemwidth that NOT includes labelwith, labelsep, only 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.

- You cannot have any standard list environments like itemize, enumerate, description, quote, quotation, verse, center, flushleft, flushright, verbatim, tabbing, trivlist, list and all environments created with \newtheorem.

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.

⋆ 4. 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 ⟨offset⟩ set by the the second optional argument. The default values for ⟨symbol⟩ and ⟨offset⟩ are `$\star$` '⋆' and the value set by `labelsep` key.

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

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

2.3.1 Keys for `\item*`

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

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

`item-pos*` = {⟨rigid length⟩}   default: *by levels*

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

2.4 The command `\item` in `enumext*`

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

```
\begin{enumext*}[widest=10,columns=4]
  \item The first
  \item* The second
  \item The third
  \item The fourth
  \item(3)* The fifth item is way too long for this and needs three columns
  \item The sixth
  \item The seventh
  \item(2)[X] The eighth item is way too long for this and needs two columns
    (\the\itemwidth)
  \item The ninth
  \item[Z] The tenth (\the\itemwidth)
\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 9. The ninth
                           two columns (196.17749pt)

Z The tenth (89.28171pt)

3 The command `\setenumext`

`\setenumext`

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

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

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

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

4 The command `\setenumextmeta`

`\setenumextmeta` `\setenumextmeta` `{⟨key name⟩}{⟨key-one = val, key-two = val, ...⟩}`
`\setenumextmeta*{⟨key name⟩}{⟨key-one = val, key-two = val, ...⟩}`
`\setenumextmeta` `[⟨enumext*⟩]{⟨key name⟩}{⟨key-one = val, key-two = val, ...⟩}`
`\setenumextmeta` `[⟨enumext, level⟩]{⟨key name⟩}{⟨key-one = val, key-two = val, ...⟩}`

The command `\setenumextmeta` adds a new *"meta-key"* for the environments `enumext` and `enumext*`, the `{⟨key name⟩}` must be different from those defined by the package. If the optional argument is not passed, the new *"meta-key"* will be created for the first level of the environment `enumext`.

The starred version `*` will create the new *"meta-key"* for the environment `enumext*` and for all levels of the environment `enumext`.

5 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.

5.1 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 \labelbx { 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={\labelbx{#1}} or wrap-label={\labelbx*{#1}}.

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

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

5.2 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.

5.2.1 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 the keyans environment the default value is 2.0pt plus 1.0pt minus 1.0pt, and for the keyans* and enumext* environments it is available but *without* effect.

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

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

> Set the *vertical space* between items, beyond the parsep. Internally sets the value of \itemsep for the current level. The default value for the first level of the environments enumext and enumext* are 4.0pt plus 2.0pt minus 1.0pt, for the rest of the levels are 2.0pt plus 1.0pt minus 1.0pt. For keyans and keyans* environments the default value is 4.0pt plus 2.0pt minus 1.0pt.

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

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

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

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

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

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

> 🎯 The following ⟨*keys*⟩ should be used with *"caution"*, they are intended to be used at the "top" and "bottom" of the environment when the columns or mini-env keys do not provide adequate *vertical spaces*. The values passed can be

rubber or *rigid* lengths, the way they are applied is the way you differ, using the *star* '*' ⟨*keys*⟩ applies `\vspace*` so that LaTeX does *not discard* this space at page break.

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

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

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

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

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

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

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

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

5.2.2 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.

5.3 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 `\begin{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*⟩}`\begin{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`: `\begin{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.

5.4 Keys for `start`, `series` and `resume`

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

Sets the *start value* of the numbering on the current level. The {⟨*integer expression*⟩} must be passed between braces, internally is evaluated and pass to the counter defined by `label` key on the current level, i.e. it is equivalent to enter `start={\dimeval{100*\value{chapter}}` or `start={100*\value{chapter}}`.

`start*` = {⟨*integer* | *string*⟩} default: *not used*

Sets the *start value* of the numbering on the current level. Internally ⟨*string*⟩ is converted and 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 `enumext*` environment and the *"first level"* of the `enumext` environment and are ignored if set when nested within 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 `start` or `start*` keys.

`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 `start` or `start*` keys.

🎯 For security reasons the `series` key will never save in {⟨*series name*⟩} the keys `series`, `resume`, `resume*`, `save-ans`, `save-key`, `start*` 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*.

5.5 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.

5.6 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`.

5.6.1 The command `\miniright`

`\miniright`

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

The `\miniright` command close the `minipage` environment on the *"left side"* and opens the `minipage` environment on the *"right side"* by starting it with the `\centering` command. It must be placed *"after"* the last `\item` of the current environment and *"before"* starting the material to be placed on the *"right side"*. The *starred argument* '*' inhibits the use of `\centering` command i.e. the usual LaTeX justification is maintained in the `minipage` on the *"right side"*.

🌑 The \footnote{⟨text⟩} command in minipage environment will work as usual. If you prefer the footnotes to be numbered (not lowercase) and outside the environment, use \footnotemark[⟨number⟩] inside the environment and \footnotetext[⟨number⟩]{⟨text⟩} outside the environment or via the after key (see §1.3.6 for full support).

5.6.2 The key mini-right

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

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

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

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

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

6 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⟩}]
    \item Text \anskey{answer}
    \item Text
      \begin{keyans}
          ...
      \end{keyans}
\end{enumext}
```

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

6.1 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.

6.1.1 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.

6.1.2 Keys for `wrap` and `display`

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

6.1.3 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*⟩}.

6.2 The command `\anskey`

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

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

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

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

6.2.1 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

6.3 The environment anskey*

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

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

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

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

6.3.1 Keys for anskey*

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

`write-env` = {⟨*file.ext*⟩}   default: *not used*

Sets the name of the ⟨*external file*⟩ in which the ⟨*contents*⟩ of the environment will be written. The ⟨*file.ext*⟩ will be created in the working directory, relative or absolute paths are not supported. If ⟨*file.ext*⟩ does not exist, it will be created or overwritten if the `overwrite` key is used.

`overwrite` = {⟨*true | false*⟩}   default: *false*

Sets whether the ⟨*file.ext*⟩ generated by `write-env` from the `anskey*` environment will be rewritten.

force-eol = { ⟨*true* | *false*⟩ }   default: *false*

Sets if the *end of line* for the ⟨*stored content*⟩ is hidden or not. This key is necessary only if the last line is the closing of some environment defined by the `fancyvrb` package as `\end{Verbatim}` or another environment that does not support a comments "%" after closing `\end{Verbatim}`%.

💣 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

6.4 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]
  \item  ⟨item content⟩
    \begin{keyans}[⟨key = val⟩]
      \item  ⟨item content⟩
      \item  [⟨custom⟩]  ⟨item content⟩
      \item*  ⟨item content⟩
      \item*[⟨content⟩]  ⟨item content⟩
    \end{keyans}
\end{enumext}
```

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

The ⟨*keys*⟩ set in the optional argument of the environment are the same (almost) as those of the `enumext` 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*`).

6.4.1 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

6.5 The environment keyanspic

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

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

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

The *starred argument* '*' places ⟨*labels*⟩ "above" the drawings or tabular, the optional argument determines the number drawings or tabular *"above"* and *"below"* within the environment. The vertical separation between *"above"* and *"below"* is controlled by the values set by `parsep` key 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*.

6.5.1 The command \anspic

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

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

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

Example

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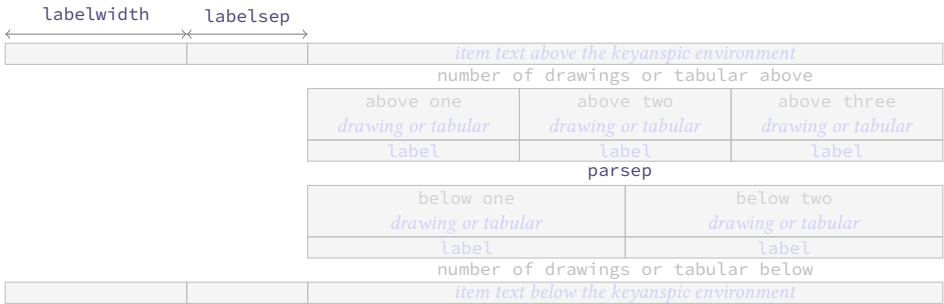
1. Question with images.

A)
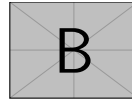
B)

C)

D)

∗ E)[note]

6.6   Printing stored content

6.6.1   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.

6.6.2   The command `\foreachkeyans`

`\foreachkeyans`

`\foreachkeyans[⟨key = val⟩]{⟨store name⟩}`

The command `\foreachkeyans` goes through and executes the command `\getkeyans` on the contents in *prop list* `{⟨store name⟩}`. If you pass without options run `\getkeyans` on all contents in *prop list* `{⟨store name⟩}`.

Options for command

`sep = {⟨code⟩}`   default: *empty*

Establishes the separation between *each* content stored in *prop list* `{⟨store name⟩}`. For example, you can use `sep={\\[10pt]}` for vertical separation of stored contents.

`step = {⟨integer⟩}`   default: *1*

Sets the increment (⟨*step*⟩) applied to the value set by key `start` for each element stored in *prop list* `{⟨store name⟩}`. The value must be a ⟨*positive integer*⟩.

`start = {⟨integer⟩}`   default: *1*

Sets the ⟨*position*⟩ of the *prop list* `{⟨store name⟩}` from which execution will start. The value must be a ⟨*positive integer*⟩.

`stop = {⟨integer⟩}`   default: *0*

Sets the ⟨*position*⟩ of the *prop list* `{⟨store name⟩}` from which execution it will finish executing. The value must be a ⟨*positive integer*⟩.

`before = {⟨code⟩}`   default: *empty*

Sets the `{⟨code⟩}` that will be executed ⟨*before*⟩ each content stored in *prop list* `{⟨store name⟩}`. The `{⟨code⟩}` must be passed between braces.

`after = {⟨code⟩}`   default: *empty*

Sets the `{⟨code⟩}` that will be executed ⟨*after*⟩ each content stored in *prop list* `{⟨store name⟩}`. The `{⟨code⟩}` must be passed between braces.

`wrapper = {⟨code {#1} more code⟩}`   default: *empty*

Wraps the content stored in *prop list* `{⟨store name⟩}` referenced by `{#1}`. The `{⟨code⟩}` must be passed between braces. For example `\foreachkeyans[wrapper={\makebox[1em][l]{#1}}]{⟨store name⟩}`.

6.6.3 The command \printkeyans

`\printkeyans` `\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   *

7 Full examples

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

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

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

Example 1

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

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

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

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

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

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

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

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

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

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

Example 2

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

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

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

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

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

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

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

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

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

1. B   *

---

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

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

Example 3

A *"simple multiple choice"* test 📄.

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

(A)

(B)

(C)

(D)

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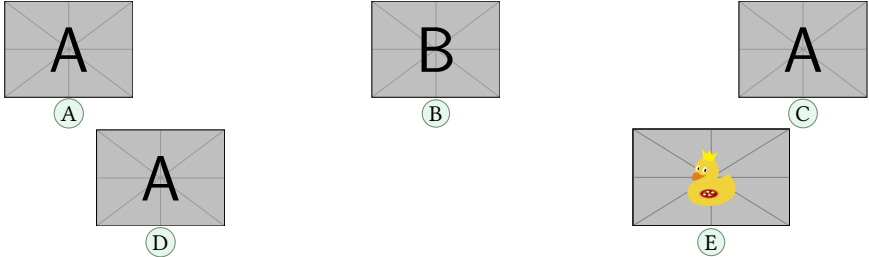
5. Question with image on left side:
   - (A) value
   - (B) value
   - (C) value
   - (D) correct
   - (E) value

Test keys

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

Example 4

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

🦆 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)

8   The way of non-enumerated lists

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

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

Fake `itemize` environment

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

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

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

Fake `description` environment

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

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

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

SomeThing A short one-line description.

   This is an entry *without* a label.

Something A short *one-line* description text.

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

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

Description indented by label

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

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

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

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

Something A short one-line description.

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

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

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

Something A short one-line description.

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

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

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

Something A short one-line description.

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

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

Description with multi-line labels

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

```
\NewDocumentCommand \labelbx { 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*={\labelbx{#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.

9   References

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

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

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

[4] GONZÁLEZ, PABLO. "scontents - Stores LaTeX contents in memory or files". Available from CTAN, https://www.ctan.org/pkg/scontents, 2022.

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

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

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

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

[9] BURNOL, JEAN-FRANÇOIS. "The footnotehyper package". Available from CTAN, https://www.ctan.org/pkg/footnotehyper, 2021.

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

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

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

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

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

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

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

10   Change history

v1.0 2024-09-25   – First public release.

11 Index of Documentation

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

C

Document class:
- article *2*
- book *2*
- exam *2*
- letter *2*
- report *2*

\columnbreak *4, 13*
\columnsep *10*

Commands provide by enumext:
- \anskey *11–13*
- \anspic *11, 12, 15*
- \foreachkeyans *16*
- \getkeyans *12, 16*
- \item* *5, 6, 8, 11, 12, 14, 15*
- \item *5, 6, 8–10, 12, 14*
- \miniright *10, 11*
- \printkeyans *6, 12, 17*
- \setenumextmeta *7*
- \setenumext *5, 6, 8, 11, 12, 14, 17*

Counters defined by enumext:
- enumXiii *4*
- enumXii *4*
- enumXiv *4*
- enumXi *4*
- enumXviii *4*
- enumXvii *4*
- enumXvi *4*
- enumXv *4*

E

Environments provide by enumext:
- anskey* *11–13*
- enumext* *4–15, 17*
- enumext *4–15, 17, 20*
- keyans* *4–9, 11–15*
- keyanspic *4, 7, 8, 11–13, 15, 20*
- keyans *4–9, 11–15, 20*

Environments:
- Verbatim *14*
- center *5*
- description *5*
- enumerate *1, 3, 5, 22*
- figure *5*
- flushleft *5*
- flushright *5*
- itemize *5*
- list *3, 5, 9, 22*
- minipage *3–5, 10, 11, 22*
- multicols *3, 4, 10*
- quotation *5*
- quote *5*
- tabbing *5*
- table *5*
- task *5*
- trivlist *5*
- verbatim *5*
- verse *5*

F

\footnote *5*

I

\itemsep *8*

K

Keys for \anskey provide by enumext:
- break-col *13*
- item-join *13*
- item-pos* *13*
- item-star *13*
- item-sym* *13*

Keys for \foreachkeyans provide by enumext:
- after *16*
- before *16*
- sep *16*
- start *16*
- step *16*
- stop *16*
- wrapper *16*

Keys for anskey* provide by enumext:
- break-col *13*
- force-eol *14*
- item-join *13*
- item-pos* *13*
- item-star *13*
- item-sym* *13*
- overwrite *13*
- write-env *13*

Keys for environments provide by enumext:
- above* *9*
- above *8, 9*
- after *9–11*
- align *7, 21*
- base-fix *8*
- before* *9*
- before *9*
- below* *9*
- below *9*
- check-ans *12*
- columns-sep *4, 10*
- columns *4, 8, 10*
- first *9*
- font *7*
- item-pos* *6*
- item-sym* *6*
- itemindent *9*
- itemsep *8*
- labelsep *3–7, 9, 10, 12, 21*
- labelwidth *3, 4, 6–10, 12, 21*
- labelwith *5*
- label *7, 8, 10, 14, 20, 21*
- list-indent *3, 9*
- list-offset *3, 9, 21*
- listparindent *9*
- mark-ans *12*
- mark-pos *12*
- mark-ref *12*
- mini-env *4, 5, 8, 10*
- mini-right* *7, 11*
- mini-right *7, 11*
- mini-sep *4, 10*
- no-store *11–13*

12   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.

12.1   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.

12.2   Initial set up

Start the DocStrip guards.

```
1  ⟨*package⟩
```

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

```
2  ⟨@@=enumext⟩
```

12.3   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    {2024-09-25}
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   }
```

12.4   Definition of variables

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

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

\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

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

Internal variables used by functions __enumext_is_not_nested:, __enumext_is_on_first_level: and __enumext_keyans_name_and_start: (§12.5.1).

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

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

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

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 (§12.10) and then modified by the function __enumext_label_style:Nnn used by label key (§12.13).

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

\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

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

Internal variables used by ref key (§12.13).

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

\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

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

Internal variables used by resume, resume* and series keys (§12.24).

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

\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

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

The variable \l__enumext_current_widest_dim stores the current label width, the variable \g__enumext_counter_styles_tl stores the default ⟨*label style*⟩ and the variable \g__enumext_widest_label_tl the label width. These variables are used by widest (§12.14) and label (§12.12) keys.

```
67 \dim_new:N \l__enumext_current_widest_dim
68 \tl_new:N  \g__enumext_counter_styles_tl
69 \tl_new:N  \g__enumext_widest_label_tl
70 \box_new:N \l__enumext_label_width_by_box
```

\l__enumext_current_widest_dim
\g__enumext_counter_styles_tl
\g__enumext_widest_label_tl
\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 (§12.17). The variables `\l__enumext_-leftmargin_X_dim` and `\l__enumext_itemindent_X_dim` are used and set by the function `\__enumext_-calc_hspace:NNNNNNNNNNNN` (§12.37.1).

```
71  \cs_set_protected:Npn \__enumext_tmp:n #1
72    {
73      \bool_new:c { l__enumext_leftmargin_tmp_#1_bool }
74      \dim_new:c  { l__enumext_leftmargin_tmp_#1_dim  }
75      \dim_new:c  { l__enumext_leftmargin_#1_dim      }
76      \dim_new:c  { l__enumext_itemindent_#1_dim      }
77    }
78  \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`
`\g__enumext_multicols_right_X_skip`
`\l__enumext_align_label_pos_X_str`

Internal variables used by `columns` key (§12.21) and `align` key (§12.12).

```
79  \cs_set_protected:Npn \__enumext_tmp:n #1
80    {
81      \skip_new:c  { l__enumext_multicols_above_#1_skip }
82      \skip_new:c  { l__enumext_multicols_below_#1_skip }
83      \skip_new:c  { g__enumext_multicols_right_#1_skip }
84      \str_new:c   { l__enumext_align_label_pos_#1_str  }
85    }
86  \clist_map_inline:nn { i, ii, iii, iv, v } { \__enumext_tmp:n {#1} }
```

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

`\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 (§12.22.4) and the keys `mini-right`, `mini-right*`, `mini-env` and `mini-sep` (§12.20, §12.22).

```
87  \int_new:N  \g__enumext_minipage_stat_int
88  \skip_new:N \l__enumext_minipage_left_skip
89  \skip_new:N \l__enumext_minipage_right_skip
90  \skip_new:N \l__enumext_minipage_after_skip
91  \skip_new:N \g__enumext_minipage_right_skip
92  \skip_new:N \g__enumext_minipage_after_skip
93  \cs_set_protected:Npn \__enumext_tmp:n #1
94    {
95      \dim_new:c  { l__enumext_minipage_left_#1_dim    }
96      \bool_new:c { l__enumext_minipage_active_#1_bool }
97    }
98  \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 bool vars `\l__enumext_wrap_label_X_bool` and `\l__enumext_wrap_label_opt_X_bool` are used by `wrap-label` and `wrap-label*` keys (§12.12), the integer `\l__enumext_start_X_int` are used by the `start` and `start*` keys (§12.14), the token list `\l__enumext_fake_item_indent_X_tl` is used by `itemindent` key (§12.17.1), the variables `\l__enumext_label_fill_left_X_tl` and `\l__enumext_-label_fill_left_X_tl` are used by the `align` key (§12.12). 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 (§12.19).

```
99   \cs_set_protected:Npn \__enumext_tmp:n #1
100    {
101      \bool_new:c { l__enumext_wrap_label_#1_bool     }
102      \bool_new:c { l__enumext_wrap_label_opt_#1_bool }
103      \int_new:c  { l__enumext_start_#1_int           }
104      \tl_new:c   { l__enumext_fake_item_indent_#1_tl }
105      \tl_new:c   { l__enumext_label_fill_left_#1_tl  }
106      \tl_new:c   { l__enumext_label_fill_right_#1_tl }
107      \bool_new:c { l__enumext_vspace_a_star_#1_bool  }
108      \bool_new:c { l__enumext_vspace_b_star_#1_bool  }
109    }
110  \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 (§12.25.1) 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 (§12.29) and the variables \l__enumext_store_anskey_env_tl and \l__enumext_store_anskey_opt_tl save the ⟨body⟩ and the ⟨keys⟩ of the environment anskey* (§12.30).

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* (§12.36) and \anspic* (§12.41.2) 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.

```
111  \bool_new:N \l__enumext_store_active_bool
112  \tl_new:N    \l__enumext_store_name_tl
113  \tl_new:N    \g__enumext_store_name_tl
114  \tl_new:N    \l__enumext_store_anskey_arg_tl
115  \tl_new:N    \l__enumext_store_anskey_env_tl
116  \tl_new:N    \l__enumext_store_anskey_opt_tl
117  \tl_new:N    \l__enumext_store_current_label_tl
118  \tl_new:N    \l__enumext_store_current_opt_arg_tl
119  \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 (§12.47).

```
120  \tl_new:N  \l__enumext_setkey_tmpa_tl
121  \tl_new:N  \l__enumext_setkey_tmpb_tl
122  \int_new:N \l__enumext_setkey_tmpa_int
123  \seq_new:N \l__enumext_setkey_tmpa_seq
124  \seq_new:N \l__enumext_setkey_tmpb_seq
```

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

Internal variables used by the \printkeyans command (§12.46) and \foreachkeyans command (§12.49).

```
125  \tl_new:N  \l__enumext_meta_path_tl
126  \seq_new:N \l__enumext_foreach_print_seq
127  \tl_new:N  \l__enumext_foreach_name_prop_tl
128  \tl_new:N  \g__enumext_foreach_default_keys_tl
```

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

Internal variables used by command \printkeyans (§12.46), show-pos key (§12.26), item-sym* key (§12.34), save-key key (§12.26.2) and *"storage level system"*.

```
129  \tl_new:N  \l__enumext_print_keyans_starred_tl
130  \str_new:N \l__enumext_mark_position_str
131  \tl_new:N  \g__enumext_item_symbol_aux_tl
132  \cs_set_protected:Npn \__enumext_tmp:n #1
133    {
134      \tl_new:c   { l__enumext_print_keyans_#1_tl         }
135      \tl_new:c   { l__enumext_store_save_key_#1_tl       }
136      \bool_new:c { l__enumext_store_save_key_#1_bool     }
137      \bool_new:c { l__enumext_store_upper_level_#1_bool  }
138    }
139  \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 and \anspic command (§12.41.1).

```
140  \seq_new:N   \l__enumext_keyans_pic_body_seq
141  \dim_new:N   \l__enumext_keyans_pic_width_dim
142  \int_new:N   \l__enumext_keyans_pic_above_int
143  \int_new:N   \l__enumext_keyans_pic_below_int
144  \bool_new:N  \l__enumext_keyans_pic_star_bool
145  \str_new:N   \l__enumext_keyans_pic_label_pos_str
146  \skip_new:N  \g__enumext_keyans_pic_parsep_skip
147  \box_new:N   \l__enumext_anspic_label_box
148  \box_new:N   \l__enumext_anspic_body_box
149  \dim_new:N   \l__enumext_anspic_label_htdp_dim
150  \dim_new:N   \l__enumext_anspic_body_htdp_dim
```

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

\l__enumext_check_answers_bool
\g__enumext_check_ans_key_bool
\l__enumext_check_start_line_env_tl
\g__enumext_check_starred_cmd_int
\g__enumext_item_anskey_int
\g__enumext_item_number_int
\g__enumext_item_number_bool
\g__enumext_item_answer_diff_int

Internal variables used by *"internal check answer"* mechanism (§12.25.3) 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.

```
151 \bool_new:N \l__enumext_check_answers_bool
152 \bool_new:N \g__enumext_check_ans_key_bool
153 \tl_new:N   \l__enumext_check_start_line_env_tl
154 \int_new:N  \g__enumext_check_starred_cmd_int
155 \int_new:N  \g__enumext_item_anskey_int
156 \int_new:N  \g__enumext_item_number_int
157 \bool_new:N \l__enumext_item_number_bool
158 \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 (§12.8). The boolean variable \l__enumext_footnotes_key_bool determine if hyperref is load with key hyperfootnotes=true.

```
159 \bool_new:N \l__enumext_hyperref_bool
160 \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 (§12.26). The variables \l__enumext_label_copy_X_tl correspond to temporary copies of the ⟨*labels*⟩ defined by level on which operations will be performed.

The variables \l__enumext_newlabel_arg_one_tl and \l__enumext_newlabel_arg_two_tl will be used to form the arguments passed to the function __enumext_newlabel:nn (§12.8) and the variable \l__enumext_write_aux_file_tl will be in charge of executing the writing code in the .aux file.

```
161 \tl_new:N \l__enumext_newlabel_arg_one_tl
162 \tl_new:N \l__enumext_newlabel_arg_two_tl
163 \tl_new:N \l__enumext_write_aux_file_tl
164 \cs_set_protected:Npn \__enumext_tmp:n #1
165   {
166     \tl_new:c { l__enumext_label_copy_#1_tl }
167   }
168 \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 (§12.42.4).

```
169 \int_new:N \g__enumext_footnote_int
170 \seq_new:N \g__enumext_footnote_arg_seq
171 \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_tmpa_X_dim
\l__enumext_item_text_X_box
\l__enumext_joined_width_X_dim
\l__enumext_item_width_X_dim
\g__enumext_item_symbol_aux_X_tl
\l__enumext_align_label_X_str
\g__enumext_minipage_active_X_bool
\l__enumext_miniright_code_X_box
\g__enumext_minipage_center_X_bool
\g__enumext_minipage_right_X_dim
\g__enumext_minipage_right_X_skip

Internal variables used by enumext* and keyans* environments.

```
172 \cs_set_protected:Npn \__enumext_tmp:n #1
173   {
174     \bool_new:c { l__enumext_item_starred_#1_bool    }
175     \int_new:c  { l__enumext_item_column_pos_#1_int  }
176     \int_new:c  { g__enumext_item_count_all_#1_int   }
177     \int_new:c  { l__enumext_joined_item_#1_int      }
178     \int_new:c  { l__enumext_joined_item_aux_#1_int  }
179     \int_new:c  { l__enumext_tmpa_#1_int             }
180     \dim_new:c  { l__enumext_tmpa_#1_dim             }
181     \box_new:c  { l__enumext_item_text_#1_box        }
182     \dim_new:c  { l__enumext_joined_width_#1_dim     }
183     \dim_new:c  { l__enumext_item_width_#1_dim       }
184     \tl_new:c   { g__enumext_item_symbol_aux_#1_tl   }
185     \str_new:c  { l__enumext_align_label_#1_str      }
186     \bool_new:c { g__enumext_minipage_active_#1_bool }
187     \box_new:c  { l__enumext_miniright_code_#1_box   }
188     \bool_new:c { g__enumext_minipage_center_#1_bool }
189     \dim_new:c  { g__enumext_minipage_right_#1_dim   }
190     \skip_new:c { g__enumext_minipage_right_#1_skip  }
191   }
192 \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`.

```
193 \clist_const:Nn \c__enumext_all_envs_clist
194   {
195     {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv},
196     {keyans}{v}, {enumext*}{vii}, {keyans*}{viii}
197   }
```

(End of definition for `\c__enumext_all_envs_clist`*.)*

12.5 Some utility functions

`\keys_precompile:neN` Non-standard kernel variants used by the `\printkeyans` command (§12.46) and `\foreachkeyans` command
`\seq_use:NV` (§12.49).

```
198 \cs_generate_variant:Nn \keys_precompile:nnN { neN }
199 \cs_generate_variant:Nn \seq_use:Nn { NV }
```

(End of definition for `\keys_precompile:neN` *and* `\seq_use:NV`*.)*

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

```
200 \cs_new_protected:Npn \__enumext_at_begin_document:n #1
201   {
202     \hook_gput_code:nnn {begindocument} {enumext} { #1 }
203   }
```

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

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

```
204 \cs_new_protected:Npn \__enumext_after_env:nn #1 #2
205   {
206     \hook_gput_code:nnn {env/#1/after} {enumext} {#2}
207   }
208 \cs_new_protected:Npn \__enumext_before_env:nn #1 #2
209   {
210     \hook_gput_code:nnn {env/#1/before} {enumext} {#2}
211   }
```

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

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

```
212 \cs_new:Nn \__enumext_level:
213   {
214     \int_to_roman:n { \l__enumext_level_int }
215   }
```

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

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

```
216 \prg_new_protected_conditional:Npnn \__enumext_if_is_int:n #1 { T, F, TF }
217   {
218     \regex_match:nnTF { ^[\+\-]?[\d]+$ } {#1} % $
219       { \prg_return_true: }
220       { \prg_return_false: }
221   }
```

(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.

```
222 \cs_new_protected:Nn \__enumext_regex_counter_style:
223   {
224     \tl_map_inline:Nn \c__enumext_counter_style_tl
225       {
226         \regex_replace_once:nnN { \c{##1}\* }
227           { \c{##1}\cB{\u{l__enumext_ref_the_count_tl}\cE} } \l__enumext_ref_key_arg_tl
228       }
229   }
```

(*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.

```
230  \cs_new:Npn \__enumext_show_length:nnn #1 #2 #3
231    {
232      * ~ #2
233      \prg_replicate:nn { 14 - \str_count:n {#2} } { ~ }
234        = ~ \use:c { #1_use:c } { l__enumext_#2_#3_#1 } \\
235    }
```

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

__enumext_unskip_unkern: The function __enumext_unskip_unkern: will remove the last ⟨*skip*⟩ or ⟨*kern*⟩ at execution time using the values 11 and 12 of \lastnodetype to apply \unskip or \unkern according to the case.

```
236  \cs_new_protected:Npn \__enumext_unskip_unkern:
237    {
238      \int_case:nnT { \lastnodetype }
239        {
240          { 11 }
241            {
242              % \typeout{SKIP} \typeout{\the\lastskip}
243              \unskip
244            }
245          { 12 }
246            {
247              % \typeout{KERN} \typeout{\the\lastkern}
248              \unkern
249            }
250        }
251    }
```

(*End of definition for* __enumext_unskip_unkern:.)

12.5.1   Utilities for environments and levels

__enumext_is_not_nested: The function __enumext_is_not_nested: set the variables \g__enumext_standar_bool and \g__-
__enumext_is_on_first_level: enumext_starred_bool to *"true"* only if the environments enumext and enumext* are nested in each other and save the environment name in \l__enumext_envir_name_tl.

```
252  \cs_new_protected:Nn \__enumext_is_not_nested:
253    {
254      \str_case:en { \@currenvir }
255        {
256          {enumext}
257            {
258              \tl_set:Nn \l__enumext_envir_name_tl { enumext }
259              \bool_lazy_and:nnT
260                { \bool_not_p:n { \g__enumext_standar_bool } }
261                { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
262                {
263                  \bool_gset_true:N \g__enumext_standar_bool
264                }
265            }
266          {enumext*}
267            {
268              \tl_set:Nn \l__enumext_envir_name_tl { enumext* }
269              \bool_lazy_and:nnT
270                { \bool_not_p:n { \g__enumext_starred_bool } }
271                { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
272                {
273                  \bool_gset_true:N \g__enumext_starred_bool
274                }
275            }
276        }
277    }
```

The function __enumext_is_on_first_level: will set the variables \l__enumext_standar_first_-
bool (§12.25.1), \l__enumext_starred_first_bool (§12.25.1) and \l__enumext_anskey_env_bool
(§12.30) 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.

```
278 \cs_new_protected:Nn \__enumext_is_on_first_level:
279   {
280     \bool_lazy_all:nT
281       {
282         { \bool_if_p:N \g__enumext_standar_bool }
283         { \int_compare_p:nNn { \l__enumext_level_int } = { 1 } }
284         { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
285       }
286       {
287         \bool_set_true:N \l__enumext_standar_first_bool
288         \bool_set_true:N \l__enumext_anskey_env_bool
289         \tl_gset:Nn \g__enumext_envir_name_tl { enumext }
290         \tl_gset:Ne \g__enumext_start_line_tl
291           {
292             on ~ line ~ \exp_not:V \inputlineno
293           }
294       }
295     \bool_lazy_all:nT
296       {
297         { \bool_if_p:N \g__enumext_starred_bool }
298         { \int_compare_p:nNn { \l__enumext_level_h_int } = { 1 } }
299         { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
300       }
301       {
302         \bool_set_true:N \l__enumext_starred_first_bool
303         \bool_set_true:N \l__enumext_anskey_env_bool
304         \tl_gset:Nn \g__enumext_envir_name_tl { enumext* }
305         \tl_gset:Ne \g__enumext_start_line_tl
306           {
307             on ~ line ~ \exp_not:V \inputlineno
308           }
309       }
310   }
```

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

__enumext_keyans_name_and_start: The function __enumext_keyans_name_and_start: will save the start line number and name of the environments keyans, keyans* and keyanspic in the variables \l__enumext_check_start_line_env_-tl and \l__enumext_envir_name_tl to use in the __enumext_check_starred_cmd:n function.

```
311 \cs_new_protected:Nn \__enumext_keyans_name_and_start:
312   {
313     \str_case:en { \@currenvir }
314       {
315         {keyans}
316           {
317             \tl_set:Nn \l__enumext_envir_name_tl { keyans }
318             \tl_set:Ne \l__enumext_check_start_line_env_tl
319               {
320                 in ~ 'keyans' ~ start ~ on ~ line ~ \exp_not:V \inputlineno
321               }
322           }
323         {keyans*}
324           {
325             \tl_set:Nn \l__enumext_envir_name_tl { keyans* }
326             \tl_set:Ne \l__enumext_check_start_line_env_tl
327               {
328                 in ~ 'keyans*' ~ start ~ on ~ line ~ \exp_not:V \inputlineno
329               }
330           }
331         {keyanspic}
332           {
333             \tl_set:Nn \l__enumext_envir_name_tl { keyanspic }
334             \tl_set:Ne \l__enumext_check_start_line_env_tl
335               {
336                 in ~ 'keyanspic' ~ start ~ on ~ line ~ \exp_not:V \inputlineno
337               }
338           }
339       }
340   }
```

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

12.5.2 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.

```
341 \cs_new_protected:Nn \__enumext_reset_global_vars:
342   {
343     \__enumext_reset_global_int:
344     \__enumext_reset_global_bool:
345     \__enumext_reset_global_tl:
346   }
347 \cs_new_protected:Nn \__enumext_reset_global_int:
348   {
349     \int_gzero:N \g__enumext_item_number_int
350     \int_gzero:N \g__enumext_item_anskey_int
351     \int_gzero:N \g__enumext_item_answer_diff_int
352   }
353 \cs_new_protected:Nn \__enumext_reset_global_bool:
354   {
355     \bool_gset_false:N \g__enumext_check_ans_key_bool
356     \bool_gset_false:N \g__enumext_standar_bool
357     \bool_gset_false:N \g__enumext_starred_bool
358   }
359 \cs_new_protected:Nn \__enumext_reset_global_tl:
360   {
361     \tl_gclear:N \g__enumext_store_name_tl
362     \tl_gclear:N \g__enumext_start_line_tl
363     \tl_gclear:N \g__enumext_envir_name_tl
364   }
```

(*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.

```
365 \cs_new_protected:Nn \__enumext_log_global_vars:
366   {
367     \msg_log:nneeee { enumext } { prop-seq-int-hook }
368       { \g__enumext_store_name_tl }
369       { \prop_count:c { g__enumext_ \g__enumext_store_name_tl _prop } }
370       { \seq_count:c { g__enumext_ \g__enumext_store_name_tl _seq } }
371       { \int_use:c { g__enumext_resume_ \g__enumext_store_name_tl _int } }
372   }
```

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.

```
373 \cs_new_protected:Nn \__enumext_log_answer_vars:
374   {
375     \msg_log:nneee { enumext } { item-answer-hook }
376       { \int_use:N \g__enumext_item_number_int }
377       { \int_use:N \g__enumext_item_anskey_int }
378       { \int_eval:n { \g__enumext_item_number_int - \g__enumext_item_anskey_int} }
379   }
```

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

12.6 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.

🏷 For compatibility with LATEX *tagged* PDF we should use \NewCommandCopy and not \cs_new_eq:NN. When *tagged* PDF is active \item is redefined using ltcmd (see latex-lab-block).

__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.

```
380 \__enumext_at_begin_document:n
```

```
381    {
382      \cs_new_eq:NN   \__enumext_start_list:nn \list
383      \cs_new_eq:NN   \__enumext_stop_list: \endlist
384      \NewCommandCopy \__enumext_item_std:w \item
385    }
```

(*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.

```
386  \__enumext_at_begin_document:n
387    {
388      \cs_new_eq:NN \__enumext_minipage:w \minipage
389      \cs_new_eq:NN \__enumext_endminipage: \endminipage
390    }
```

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

12.7 The internal minipage environment

__enumext_internal_mini_page:
__enumext_mini_env*

The function __enumext_internal_mini_page: creates a internal __enumext_mini_page 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 \skip_vertical:N \c_zero_skip to maintain alignment on *"top"* in the first part and \skip_vertical:N \c_zero_skip in the second part to allow spaces *"below"*. 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 (§12.38) and __enumext_safe_exec_vii: in the enumext* environment definition (§12.43)

```
391  \cs_new_protected:Nn \__enumext_internal_mini_page:
392    {
393      \int_compare:nNnT { \l__enumext_level_int } = { 0 }
394        {
395          \DeclareDocumentEnvironment{__enumext_mini_page}{ m }
396            {
397              \__enumext_minipage:w [ t ] { ##1 }
398                \legacy_if_gset_false:n { @minipage }
399                \skip_vertical:N \c_zero_skip
400            }
401            {
402                \skip_vertical:N \c_zero_skip
403              \__enumext_endminipage:
404            }
405        }
406    }
```

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

12.8 Compatibility with hyperref and footnotehyper

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

```
407  \hook_gput_code:nnn { begindocument } { enumext } { \__enumext_after_hyperref: }
408  \hook_gset_rule:nnnn { begindocument } { enumext } { after } { hyperref }
```

__enumext_after_hyperref:
__enumext_hypertarget:nn
__enumext_phantomsection:

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

```
409  \cs_new_protected:Nn \__enumext_after_hyperref:
410    {
411      \IfPackageLoadedTF { hyperref }
412        {
413          \msg_info:nnn { enumext } { package-load } { hyperref }
414          \bool_set_true:N \l__enumext_hyperref_bool
```

```
415        \IfHyperBoolean{hyperfootnotes}
416          {
417            % \typeout{hyperfootnotes=true}
418            \bool_set_true:N \l__enumext_footnotes_key_bool
419          }
420          {
421            % \typeout{hyperfootnotes=false}
422          }
423        }
424      {  }
```

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`.

```
425      \bool_if:NT \l__enumext_footnotes_key_bool
426        {
427          \IfPackageLoadedTF { footnotehyper }
428            {
429              \msg_info:nnn { enumext } { package-load } { footnotehyper }
430            }
431            {
432              % \typeout{No ~ footnotehyper ~ load}
433              % \typeout{Load ~ and  ~ use  ~ \string\makesavenoteenv{enumext*}}
434              \bool_set_false:N \l__enumext_footnotes_key_bool
435            }
436        }
```

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.

```
437      \bool_if:NTF \l__enumext_hyperref_bool
438        {
439          \cs_new_eq:NN \__enumext_hypertarget:nn \hypertarget
440          \cs_new_eq:NN \__enumext_phantomsection: \phantomsection
441        }
442        {
443          \cs_new_eq:NN \__enumext_hypertarget:nn \use_none:nn
444          \cs_new_eq:NN \__enumext_phantomsection: \prg_do_nothing:
445        }
446    }
```

(*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.

```
447    \cs_new_protected:Npn \__enumext_newlabel:nn #1 #2
448      {
449        \protected@write \@auxout { }
450          {
451            \token_to_str:N \newlabel {#1}
452              {
453                {#2}
454                \bool_if:NT \l__enumext_hyperref_bool
455                  { { \thepage } {#2} {#1} }
456                { }
457              }
458          }
459        \__enumext_hypertarget:nn {#1} { }
460        \__enumext_phantomsection:
461      }
```

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

12.9 Definition of public dimension

The package enumext only provides a single public dimension \itemwidth and is intended for user convenience only and is not for internal use as such. This dimension is set in all environments and is only used by the wrap-ans key at its default value.

```
462 \dim_zero_new:N \itemwidth
```

12.10 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.

```
463 \cs_new_protected:Npn \__enumext_define_counters:Nn #1 #2
464   {
465     \cs_if_exist:cTF { c@ #2 }
466       { \msg_fatal:nnn { enumext } { counters }{ #2 } }
467       {
468         \tl_set:Nn #1 { #2 }
469         \newcounter { #2 }
470       }
471   }
```

(*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.

```
472 \__enumext_define_counters:Nn \l__enumext_counter_i_tl    { enumXi     }
473 \__enumext_define_counters:Nn \l__enumext_counter_ii_tl   { enumXii    }
474 \__enumext_define_counters:Nn \l__enumext_counter_iii_tl  { enumXiii   }
475 \__enumext_define_counters:Nn \l__enumext_counter_iv_tl   { enumXiv    }
476 \__enumext_define_counters:Nn \l__enumext_counter_v_tl    { enumXv     }
477 \__enumext_define_counters:Nn \l__enumext_counter_vi_tl   { enumXvi    }
478 \__enumext_define_counters:Nn \l__enumext_counter_vii_tl  { enumXvii   }
479 \__enumext_define_counters:Nn \l__enumext_counter_viii_tl { enumXviii  }
```

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

12.11 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.

```
480 \cs_new_protected:Npn \__enumext_register_counter_style:Nn #1 #2
481   {
482     \tl_const:cn { c__enumext_widest_ \cs_to_str:N #1 _tl } {#2}
483     \tl_gput_right:Nn \g__enumext_counter_styles_tl {#1}
484   }
485 \__enumext_register_counter_style:Nn \arabic { 0 }
486 \__enumext_register_counter_style:Nn \Alph   { M }
487 \__enumext_register_counter_style:Nn \alph   { m }
488 \__enumext_register_counter_style:Nn \Roman  { VIII }
489 \__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.

```
490 \cs_new_protected:Npn \__enumext_label_width_by_box:Nn #1 #2
491   {
492     \hbox_set:Nn \l__enumext_label_width_by_box {#2}
493     \dim_set:Nn #1 { \box_wd:N \l__enumext_label_width_by_box }
494   }
495 \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.

```
496 \cs_new_protected:Npn \__enumext_label_style:Nnn #1 #2 #3
497   {
498     \tl_clear_new:N #1
499     \tl_put_right:Ne #1 { \tl_trim_spaces:n {#3} }
500     \tl_gset_eq:NN \g__enumext_widest_label_tl #1
501     \tl_map_inline:Nn \g__enumext_counter_styles_tl
502       {
503         \tl_replace_all:Nne #1 { ##1* } { \exp_not:N ##1 {#2} }
504         \tl_greplace_all:Nne \g__enumext_widest_label_tl { ##1* }
505           { \tl_use:c { c__enumext_widest_ \cs_to_str:N ##1 _tl } }
506       }
507     \__enumext_label_width_by_box:Nn \l__enumext_current_widest_dim
508       { \tl_use:N \g__enumext_widest_label_tl }
509     \tl_set_eq:cN { the #2 } #1
510   }
511 \cs_generate_variant:Nn \__enumext_label_style:Nnn { cvn }
```

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

12.12 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.

```
512 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
513   {
514     \keys_define:nn { enumext / #1 }
515       {
516         font        .tl_set:c   = { l__enumext_label_font_style_#2_tl },
517         font        .value_required:n = true,
518         labelsep    .dim_set:c  = { l__enumext_labelsep_#2_dim },
519         labelsep    .initial:n  = {0.3333em},
520         labelsep    .value_required:n = true,
521         labelwidth  .dim_set:c  = { l__enumext_labelwidth_#2_dim },
522         labelwidth  .value_required:n = true,
523         wrap-label  .cs_set_protected:cp = { __enumext_wrapper_label_#2:n } ##1,
524         wrap-label  .initial:n  = {##1},
525         wrap-label  .value_required:n = true,
526         wrap-label* .code:n = {
527                               \bool_set_true:c { l__enumext_wrap_label_opt_#2_bool }
528                               \keys_set:nn { enumext / #1 } { wrap-label = {##1} }
529                             },
530         wrap-label* .value_required:n = true,
531       }
532   }
533 \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.

```
534 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
535   {
536     \keys_define:nn { enumext / #1 }
537       {
538         align .choice:,
539         align / left    .code:n =
540                           {
541                             \tl_clear:c { l__enumext_label_fill_left_#2_tl  }
542                             \tl_set:cn  { l__enumext_label_fill_right_#2_tl } { \hfill }
543                             \str_set:cn { l__enumext_align_label_pos_#2_str } { l }
544                           },
545         align / right   .code:n =
546                           {
547                             \tl_set:cn  { l__enumext_label_fill_left_#2_tl  } { \hfill }
```

```
548                                    \tl_clear:c { l__enumext_label_fill_right_#2_tl }
549                                    \str_set:cn { l__enumext_align_label_pos_#2_str } { r }
550                                  },
551          align / center  .code:n =
552                                  {
553                                    \tl_set:cn { l__enumext_label_fill_left_#2_tl  } { \hfill }
554                                    \tl_set:cn { l__enumext_label_fill_right_#2_tl } { \hfill }
555                                    \str_set:cn { l__enumext_align_label_pos_#2_str } { c }
556                                  },
557          align / unknown .code:n =
558                                  \msg_error:nneee { enumext } { unknown-choice }
559                                    { align } { left, ~ right, ~  center } { \exp_not:n {##1} },
560          align .initial:n  = left,
561          align .value_required:n  = true,
562        }
563   }
564 \clist_map_inline:nn
565   {
566     {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {keyans}{v}
567   }
568   { \__enumext_tmp:nn #1 }
```

🏷 For compatibility with LᴬTEX *tagged* PDF we must set `\l__enumext_align_label_pos_X_str`. When *tagged* PDF is active `\makelabel` is redefined and the only way to get the align key to work correctly is by using `\makebox`.

```
569 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
570   {
571     \keys_define:nn { enumext / #1 }
572       {
573         align .choice:,
574         align / left    .code:n = \str_set:cn { l__enumext_align_label_#2_str } { l },
575         align / right   .code:n = \str_set:cn { l__enumext_align_label_#2_str } { r },
576         align / center  .code:n = \str_set:cn { l__enumext_align_label_#2_str } { c },
577         align / unknown .code:n =
578                              \msg_error:nneee { enumext } { unknown-choice }
579                                { align } { left, ~ right, ~  center } { \exp_not:n {##1} },
580         align .initial:n = left,
581         align .value_required:n = true,
582       }
583   }
584 \clist_map_inline:nn { {enumext*}{vii}, {keyans*}{viii} } { \__enumext_tmp:nn #1 }
```

(*End of definition for* align.)

12.13   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.

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

label
ref
`\l__enumext_label_i_tl`
`\l__enumext_label_ii_tl`
`\l__enumext_label_iii_tl`
`\l__enumext_label_iv_tl`

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

```
585 \cs_set_protected:Npn \__enumext_tmp:nnn #1 #2 #3
586   {
587     \keys_define:nn { enumext / #1 }
588       {
589         label .code:n    = {
590                             \__enumext_label_style:cvn { l__enumext_label_#2_tl }
591                               { l__enumext_counter_#2_tl } {##1}
592                             \dim_set_eq:cN  { l__enumext_labelwidth_#2_dim }
593                               \l__enumext_current_widest_dim
594                            },
595         label .initial:n = #3,
596         label .value_required:n = true,
597         ref   .code:n    = \__enumext_standar_ref:n {##1},
598         ref   .value_required:n = true,
599       }
600   }
601 \__enumext_tmp:nnn { level-1 } {   i } { \arabic*.}
602 \__enumext_tmp:nnn { level-2 } {  ii } { (\alph*) }
603 \__enumext_tmp:nnn { level-3 } { iii } { \roman*. }
604 \__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.

```
605 \cs_new_protected:Npn \__enumext_standar_ref:n #1
606   {
607     \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
608     \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
609       {
610         \msg_error:nnn { enumext } { key-ref-empty } { enumext }
611       }
612       {
613         \tl_set_eq:Nc
614           \l__enumext_ref_the_count_tl { l__enumext_counter_ \__enumext_level: _tl }
615         \__enumext_regex_counter_style:
616         \tl_set_eq:Nc
617           \l__enumext_ref_the_count_tl { l__enumext_the_counter_ \__enumext_level: _tl }
618         \tl_put_right:ce { l__enumext_renew_the_count_ \__enumext_level: _tl }
619           {
620             \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl }
621               { \exp_not:V \l__enumext_ref_key_arg_tl }
622           }
623       }
624   }
```

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

```
625 \cs_new_protected:Nn \__enumext_standar_ref:
626   {
627     \tl_if_empty:cF { l__enumext_renew_the_count_ \__enumext_level: _tl }
628       {
629         \tl_use:c { l__enumext_renew_the_count_ \__enumext_level: _tl }
630       }
631   }
```

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

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

label
ref
\l__enumext_label_vii_tl
\l__enumext_label_viii_tl

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

```
632 \cs_set_protected:Npn \__enumext_tmp:nnn #1 #2 #3
633   {
634     \keys_define:nn { enumext / #1 }
635       {
636         label .code:n    = {
637                             \__enumext_label_style:cvn { l__enumext_label_#2_tl }
638                             { l__enumext_counter_#2_tl } {##1}
639                             \dim_set_eq:cN  { l__enumext_labelwidth_#2_dim }
640                             \l__enumext_current_widest_dim
641                           },
642         label .initial:n = #3,
643         label .value_required:n = true,
644         ref   .code:n    = \__enumext_starred_ref:n {##1},
645         ref   .value_required:n = true,
646       }
647   }
648 \__enumext_tmp:nnn { enumext* } {  vii } { \arabic*.}
649 \__enumext_tmp:nnn { keyans*  } { viii } { \Alph*) }
```

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

__enumext_starred_ref:n
__enumext_starred_ref:

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

```
650 \cs_new_protected:Npn \__enumext_starred_ref:n #1
651   {
652     \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
653     \int_compare:nNnT { \l__enumext_level_h_int } = { 1 }
654       {
```

```
655        \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
656          {
657            \msg_error:nnn { enumext } { key-ref-empty } { enumext* }
658          }
659          {
660            \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_counter_vii_tl
661            \__enumext_regex_counter_style:
662            \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_the_counter_vii_tl
663            \tl_put_right:Ne \l__enumext_renew_the_count_vii_tl
664              {
665                \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl }
666                  { \exp_not:V \l__enumext_ref_key_arg_tl }
667              }
668          }
669        }
670      \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
671        {
672          \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
673            {
674              \msg_error:nnn { enumext } { key-ref-empty } { keyans* }
675            }
676            {
677              \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_counter_viii_tl
678              \__enumext_regex_counter_style:
679              \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_the_counter_viii_tl
680              \tl_put_right:Ne \l__enumext_renew_the_count_viii_tl
681                {
682                  \exp_not:N \renewcommand  { \exp_not:V \l__enumext_ref_the_count_tl }
683                    { \exp_not:V \l__enumext_ref_key_arg_tl }
684                }
685            }
686        }
687    }
```

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.

```
688  \cs_new_protected:Nn \__enumext_starred_ref:
689    {
690      \int_compare:nNnT { \l__enumext_level_h_int } = { 1 }
691        {
692          \tl_if_empty:NF \l__enumext_renew_the_count_vii_tl
693            {
694              \tl_use:N \l__enumext_renew_the_count_vii_tl
695            }
696        }
697      \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
698        {
699          \tl_if_empty:NF \l__enumext_renew_the_count_viii_tl
700            {
701              \tl_use:N \l__enumext_renew_the_count_viii_tl
702            }
703        }
704    }
```

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

12.13.3 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.

```
705  \keys_define:nn { enumext / keyans }
706    {
707      label .code:n    = {
708                          \__enumext_label_style:cvn { l__enumext_label_v_tl }
709                            { l__enumext_counter_v_tl } {#1}
710                          \dim_set_eq:cN  { l__enumext_labelwidth_v_dim }
711                            \l__enumext_current_widest_dim
712                          \__enumext_label_style:cvn { l__enumext_label_vi_tl }
713                            { l__enumext_counter_vi_tl } {#1}
714                          \dim_set_eq:cN  { l__enumext_labelwidth_v_dim }
715                            \l__enumext_current_widest_dim
716                        },
```

```
717     label  .initial:n = \Alph*),
718     label  .value_required:n = true,
719     ref    .code:n    = \__enumext_keyans_ref:n {#1},
720     ref    .value_required:n = true,
721   }
```

(*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.

```
722 \cs_new_protected:Npn \__enumext_keyans_ref:n #1
723   {
724     \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
725     \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
726       {
727         \msg_error:nnn { enumext } { key-ref-empty } { keyans }
728       }
729       {
730         \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_counter_v_tl
731         \__enumext_regex_counter_style:
732         \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_the_counter_v_tl
733         \tl_put_right:Ne \l__enumext_renew_the_count_v_tl
734           {
735             \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl }
736               { \exp_not:V \l__enumext_ref_key_arg_tl }
737           }
738       }
739   }
```

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

```
740 \cs_new_protected:Nn \__enumext_keyans_ref:
741   {
742     \tl_if_empty:NF \l__enumext_renew_the_count_v_tl
743       {
744         \tl_use:N \l__enumext_renew_the_count_v_tl
745       }
746   }
```

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

12.14 Setting start, start* and widest keys

__enumext_start_from:NNn
__enumext_start_from:ccn
__enumext_start_from:cce

The function __enumext_start_from:NNn used by start and start* keys 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.

```
747 \cs_new_protected:Npn \__enumext_start_from:NNn #1 #2 #3
748   {
749     \__enumext_if_is_int:nTF { #3 }
750       {
751         \int_set:Nn #2 {#3}
752       }
753       {
754         \regex_match:nVT { \c{Alph} | \c{alph} } {#1}
755           { \int_set:Nn #2 { \int_from_alph:n {#3} } }
756         \regex_match:nVT { \c{Roman} | \c{roman} } {#1}
757           { \int_set:Nn #2  { \int_from_roman:n {#3} } }
758       }
759   }
760 \cs_generate_variant:Nn \__enumext_start_from:NNn { ccn, cce }
```

(*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.

```
761  \cs_new_protected:Npn \__enumext_widest_from:nNNn #1 #2 #3 #4
762    {
763      \__enumext_if_is_int:nTF {#4}
764        {
765          \setcounter{enumX#1} { #4 }
766        }
767        {
768          \regex_match:nVT { \c{Alph} | \c{alph} } {#2}
769            { \setcounter{enumX#1} { \int_from_alph:n {#4} } }
770          \regex_match:nVT { \c{Roman} | \c{roman} } {#2}
771            { \setcounter{enumX#1} { \int_from_roman:n {#4} } }
772        }
773      \__enumext_label_width_by_box:cv
774        { l__enumext_labelwidth_#1_dim } { l__enumext_label_#1_tl }
775    }
776  \cs_generate_variant:Nn \__enumext_widest_from:nNNn { nccn }
```

(*End of definition for* `\__enumext_widest_from:nNNn`.)

start
start*
widest
Now define and set `start*`, `start` and `widest` keys for `enumext`, `enumext*`, `keyans` and `keyans*` environments.

```
777  \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
778    {
779      \keys_define:nn { enumext / #1 }
780        {
781          start* .code:n    = {
782                                \__enumext_start_from:ccn
783                                  { l__enumext_label_#2_tl }
784                                  { l__enumext_start_#2_int } {##1}
785                              },
786          start* .value_required:n = true,
787          start  .code:n    = {
788                                \__enumext_start_from:cce
789                                  { l__enumext_label_#2_tl }
790                                  { l__enumext_start_#2_int } { \int_eval:n {##1} }
791                              },
792          start  .initial:n = 1,
793          start  .value_required:n = true,
794          widest .code:n    = {
795                                \__enumext_widest_from:nccn {#2}
796                                  { l__enumext_label_#2_tl }
797                                  { l__enumext_labelwidth_#2_dim } {##1}
798                              },
799          widest .value_required:n = true,
800        }
801    }
802  \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }
```

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

12.15   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.

```
803  \cs_set_protected:Npn \__enumext_tmp:nnnnnn #1 #2 #3 #4 #5 #6
804    {
805      \keys_define:nn { enumext / #1 }
806        {
807          topsep     .skip_set:c = { l__enumext_topsep_#2_skip },
808          topsep     .initial:n  = {#3},
809          topsep     .value_required:n = true,
810          partopsep .skip_set:c = { l__enumext_partopsep_#2_skip },
811          partopsep .initial:n  = {#4},
812          partopsep .value_required:n = true,
813          parsep     .skip_set:c = { l__enumext_parsep_#2_skip },
814          parsep     .initial:n  = {#5},
815          parsep     .value_required:n = true,
```

```
816          itemsep   .skip_set:c = { l__enumext_itemsep_#2_skip },
817          itemsep   .initial:n  = {#6},
818          itemsep   .value_required:n = true,
819          noitemsep .meta:n      = { itemsep = 0pt, parsep = 0pt },
820          noitemsep .value_forbidden:n = true,
821          nosep     .meta:n      = {
822                                   itemsep = 0pt, parsep= 0pt,
823                                   topsep = 0pt, partopsep = 0pt,
824                                  },
825          nosep     .value_forbidden:n = true,
826        }
827    }
```

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

```
828 \__enumext_tmp:nnnnnn { level-1 } { i } { 8.0pt plus 2.0pt minus 4.0pt }
829    { 2.0pt plus 1.0pt minus 1.0pt } { 4.0pt plus 2.0pt minus 1.0pt }
830    { 4.0pt plus 2.0pt minus 1.0pt }
831 \__enumext_tmp:nnnnnn { level-2 } { ii } { 4.0pt plus 2.0pt minus 1.0pt }
832    { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
833    { 2.0pt plus 1.0pt minus 1.0pt }
834 \__enumext_tmp:nnnnnn { level-3 } { iii } { 2.0pt plus 1.0pt minus 1.0pt }
835    { 1.0pt minus 1.0pt }{ 0pt }{ 2.0pt plus 1.0pt minus 1.0pt }
836 \__enumext_tmp:nnnnnn { level-4 } { iv } { 2.0pt plus 1.0pt minus 1.0pt }
837    { 1.0pt minus 1.0pt }{ 0pt }{ 2.0pt plus 1.0pt minus 1.0pt }
838 \__enumext_tmp:nnnnnn { keyans  } { v }{ 4.0pt plus 2.0pt minus 1.0pt }
839    { 2.0pt plus 1.0pt minus 1.0pt }{ 2.0pt plus 1.0pt minus 1.0pt }
840    { 2.0pt plus 1.0pt minus 1.0pt }
841 \__enumext_tmp:nnnnnn { enumext* } { vii } { 8.0pt plus 2.0pt minus 4.0pt }
842    { 2.0pt plus 1.0pt minus 1.0pt } { 4.0pt plus 2.0pt minus 1.0pt }
843    { 4.0pt plus 2.0pt minus 1.0pt }
844 \__enumext_tmp:nnnnnn { keyans* } { viii } { 4.0pt plus 2.0pt minus 1.0pt }
845    { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
846    { 2.0pt plus 1.0pt minus 1.0pt }
```

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

12.16   Setting base-fix key

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

base-fix
__enumext_nested_base_line_fix:

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

```
847 \cs_set_protected:Npn \__enumext_tmp:n #1
848   {
849     \keys_define:nn { enumext / #1 }
850       {
851         base-fix .bool_set:N = \l__enumext_base_line_fix_bool,
852         base-fix .initial:n  = false,
853         base-fix .value_forbidden:n = true,
854       }
855   }
856 \clist_map_inline:nn { level-1, enumext* } { \__enumext_tmp:n {#1} }
```

The function __enumext_nested_base_line_fix: will be in charge of applying the baseline correction and adjusting the ⟨*keys*⟩. This function is passed to the function __enumext_parse_keys:n in the enumext environment definition (§12.38) and to the function __enumext_parse_keys_vii:n in the enumext* environment definition (§12.43)

🎯 This key is enabled by default in the command \printkeyans (§12.46).

```
857 \cs_new_protected:Nn \__enumext_nested_base_line_fix:
858   {
859     \bool_lazy_and:nnT
860       { \bool_if_p:N \l__enumext_standar_first_bool }
861       { \bool_if_p:N \l__enumext_base_line_fix_bool }
862       {
863         \mode_leave_vertical:
864         \vspace { -\baselineskip }
865         \keys_set:nn { enumext / level-1 }
866           {
867             topsep = 0pt, above = 0pt, above* = 0pt,
868           }
```

```
869          }
870      \bool_lazy_and:nnT
871        { \bool_if_p:N \l__enumext_starred_first_bool }
872        { \bool_if_p:N \l__enumext_base_line_fix_bool }
873        {
874          \mode_leave_vertical:
875          \vspace { -\baselineskip }
876          \keys_set:nn { enumext / enumext* }
877            {
878              topsep = 0pt, above = 0pt, above* = 0pt,
879            }
880        }
881      \bool_set_false:N \l__enumext_base_line_fix_bool
882    }
```

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

12.17 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.

```
883  \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
884    {
885      \keys_define:nn { enumext / #1 }
886        {
887          itemindent    .dim_set:c = { l__enumext_fake_item_indent_#2_dim },
888          itemindent    .value_required:n = true,
889          rightmargin   .dim_set:c = { l__enumext_rightmargin_#2_dim },
890          rightmargin   .value_required:n = true,
891          listparindent .dim_set:c = { l__enumext_listparindent_#2_dim },
892          listparindent .value_required:n = true,
893          list-offset   .dim_set:c = { l__enumext_listoffset_#2_dim },
894          list-offset   .value_required:n = true,
895          list-indent   .code:n      =
896                          \bool_set_true:c { l__enumext_leftmargin_tmp_#2_bool }
897                          \dim_set:cn { l__enumext_leftmargin_tmp_#2_dim } {##1},
898          list-indent   .value_required:n = true,
899        }
900    }
901  \clist_map_inline:nn
902    {
903      {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {keyans}{v}
904    }
905    { \__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.

```
906  \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
907    {
908      \keys_define:nn { enumext / #1 }
909        {
910          itemindent    .dim_set:c = { l__enumext_fake_item_indent_#2_dim },
911          itemindent    .value_required:n = true,
912          rightmargin   .dim_set:c = { l__enumext_rightmargin_#2_dim },
913          rightmargin   .value_required:n = true,
914          listparindent .dim_set:c = { l__enumext_listparindent_#2_dim },
915          listparindent .value_required:n = true,
916          list-offset   .dim_set:c = { l__enumext_listoffset_#2_dim },
917          list-offset   .value_required:n = true,
918          list-indent   .meta:n     = { list-offset = ##1 },
919          list-indent   .value_required:n = true,
920        }
921    }
922  \clist_map_inline:nn
923    {
924      {enumext*}{vii}, {keyans*}{viii}
925    }
926    { \__enumext_tmp:nn #1 }
```

12.17.1 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.

```
927  \cs_set_protected:Nn \__enumext_fake_item:
928    {
929      \dim_compare:nNnT
930        { \dim_use:c { l__enumext_fake_item_indent_ \__enumext_level: _dim } }
931        >
932        { \c_zero_dim }
933        {
934          \tl_set:ce { l__enumext_fake_item_indent_ \__enumext_level: _tl }
935            {
936              \exp_not:N \mode_leave_vertical:
937              \exp_not:n { \skip_horizontal:n }
938                { \dim_use:c { l__enumext_fake_item_indent_ \__enumext_level: _dim } }
939              \ignorespaces
940            }
941        }
942    }
943  \cs_set_protected:Nn \__enumext_keyans_fake_item:
944    {
945      \dim_compare:nNnT
946        { \l__enumext_fake_item_indent_v_dim } > { \c_zero_dim }
947        {
948          \tl_set:Ne \l__enumext_fake_item_indent_v_tl
949            {
950              \exp_not:N \mode_leave_vertical:
951              \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_v_dim
952              \ignorespaces
953            }
954        }
955    }
956  \cs_set_protected:Nn \__enumext_fake_item_vii:
957    {
958      \dim_compare:nNnT
959        { \l__enumext_fake_item_indent_vii_dim } > { \c_zero_dim }
960        {
961          \tl_set:Ne \l__enumext_fake_item_indent_vii_tl
962            {
963              \exp_not:N \mode_leave_vertical:
964              \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_vii_dim
965              \ignorespaces
966            }
967        }
968    }
969  \cs_set_protected:Nn \__enumext_fake_item_viii:
970    {
971      \dim_compare:nNnT
972        { \l__enumext_fake_item_indent_viii_dim } > { \c_zero_dim }
973        {
974          \tl_set:Ne \l__enumext_fake_item_indent_viii_tl
975            {
976              \exp_not:N \mode_leave_vertical:
977              \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_viii_dim
978              \ignorespaces
979            }
980        }
981    }
```

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

12.18 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.

```
982  \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
983    {
```

```
984      \keys_define:nn { enumext / #1 }
985        {
986          show-length .bool_set:c = { l__enumext_show_length_#2_bool },
987          show-length .initial:n  = false,
988        }
989    }
990  \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }
```

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

12.19   Setting before, after and first keys

before
before*
after
first

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

```
991  \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
992    {
993      \keys_define:nn { enumext / #1 }
994        {
995          before  .tl_set:c   = { l__enumext_before_no_starred_key_#2_tl },
996          before  .value_required:n = true,
997          before* .tl_set:c   = { l__enumext_before_starred_key_#2_tl },
998          before* .value_required:n = true,
999          after   .tl_set:c   = { l__enumext_after_stop_list_#2_tl },
1000          after   .value_required:n = true,
1001          first   .tl_set:c   = { l__enumext_after_list_args_#2_tl },
1002          first   .value_required:n = true,
1003        }
1004    }
1005  \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }
```

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

12.19.1   Functions for before, after and first keys in enumext

__enumext_before_args_exec:
__enumext_before_keys_exec:
__enumext_after_stop_list:
__enumext_after_args_exec:

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

```
1006  \cs_new_protected:Nn \__enumext_before_args_exec:
1007    {
1008      \tl_use:c { l__enumext_before_starred_key_ \__enumext_level: _tl }
1009    }
```

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*⟩: \list{⟨*arg one*⟩}{⟨*arg two*⟩{⟨*code*⟩}}

```
1010  \cs_new_protected:Nn \__enumext_before_keys_exec:
1011    {
1012      \tl_use:c { l__enumext_before_no_starred_key_ \__enumext_level: _tl }
1013    }
```

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

```
1014  \cs_new_protected:Nn \__enumext_after_stop_list:
1015    {
1016      \tl_use:c { l__enumext_after_stop_list_ \__enumext_level: _tl }
1017    }
```

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: \list{⟨*arg one*⟩}{⟨*arg two*⟩}{⟨*code*⟩}\item.

```
1018  \cs_new_protected:Nn \__enumext_after_args_exec:
1019    {
1020      \tl_use:c { l__enumext_after_list_args_ \__enumext_level: _tl }
1021    }
```

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

12.19.2 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:`

Same implementation as the one used in the enumext environment.

```
1022 \cs_new_protected:Nn \__enumext_before_args_exec_v:
1023   {
1024     \tl_use:N \l__enumext_before_starred_key_v_tl
1025   }
1026 \cs_new_protected:Nn \__enumext_before_keys_exec_v:
1027   {
1028     \tl_use:N \l__enumext_before_no_starred_key_v_tl
1029   }
1030 \cs_new_protected:Nn \__enumext_after_stop_list_v:
1031   {
1032     \tl_use:N \l__enumext_after_stop_list_v_tl
1033   }
1034 \cs_new_protected:Nn \__enumext_after_args_exec_v:
1035   {
1036     \tl_use:N \l__enumext_after_list_args_v_tl
1037   }
```

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

12.19.3 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:`

Same implementation as the one used in the enumext environment.

```
1038 \cs_new_protected:Nn \__enumext_before_args_exec_vii:
1039   {
1040     \tl_use:N \l__enumext_before_starred_key_vii_tl
1041   }
1042 \cs_new_protected:Nn \__enumext_before_args_exec_viii:
1043   {
1044     \tl_use:N \l__enumext_before_starred_key_viii_tl
1045   }
1046 \cs_new_protected:Nn \__enumext_before_keys_exec_vii:
1047   {
1048     \tl_use:N \l__enumext_before_no_starred_key_vii_tl
1049   }
1050 \cs_new_protected:Nn \__enumext_before_keys_exec_viii:
1051   {
1052     \tl_use:N \l__enumext_before_no_starred_key_viii_tl
1053   }
1054 \cs_new_protected:Nn \__enumext_after_stop_list_vii:
1055   {
1056     \tl_use:N \l__enumext_after_stop_list_vii_tl
1057   }
1058 \cs_new_protected:Nn \__enumext_after_stop_list_viii:
1059   {
1060     \tl_use:N \l__enumext_after_stop_list_viii_tl
1061   }
1062 \cs_new_protected:Nn \__enumext_after_args_exec_vii:
1063   {
1064     \tl_use:N \l__enumext_after_list_args_vii_tl
1065   }
1066 \cs_new_protected:Nn \__enumext_after_args_exec_viii:
1067   {
1068     \tl_use:N \l__enumext_after_list_args_viii_tl
1069   }
```

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

12.20 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.

```
1070 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1071   {
1072     \keys_define:nn { enumext / #1 }
1073       {
1074         mini-env    .dim_set:c  = { l__enumext_minipage_right_#2_dim },
1075         mini-env    .value_required:n = true,
1076         mini-sep    .dim_set:c  = { l__enumext_minipage_hsep_#2_dim },
```

```
1077        mini-sep    .initial:n  = 0.3333em,
1078        mini-sep    .value_required:n = true,
1079        columns-sep .dim_set:c  = { l__enumext_columns_sep_#2_dim },
1080        columns-sep .value_required:n = true,
1081        columns     .int_set:c  = { l__enumext_columns_#2_int },
1082        columns     .initial:n  = 1,
1083        columns     .value_required:n = true,
1084      }
1085    }
1086 \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.

```
1087 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1088   {
1089     \keys_define:nn { enumext / #1 }
1090       {
1091         mini-right  .tl_gset:c = { g__enumext_miniright_code_#2_tl },
1092         mini-right  .value_required:n = true,
1093         mini-right* .code:n    = {
1094                                   \bool_gset_true:c { g__enumext_minipage_center_#2_bool }
1095                                   \keys_set:nn { enumext / #1 } { mini-right = {##1} }
1096                                 },
1097         mini-right* .value_required:n = true,
1098       }
1099   }
1100 \clist_map_inline:nn { {enumext*}{vii}, {keyans*}{viii} } { \__enumext_tmp:nn #1 }
```

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

12.21   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.

12.21.1   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*.

```
1101 \cs_new_protected:Nn \__enumext_multi_set_vskip:
1102   {
1103     \skip_set:cn { l__enumext_multicols_above_ \__enumext_level: _skip }
1104       {
1105         \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
1106       }
1107     \skip_set:cn { l__enumext_multicols_below_ \__enumext_level: _skip }
1108       {
1109         \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
1110       }
```

```
1111        \__enumext_add_pre_parsep:
1112      }
```

(*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*.

```
1113  \cs_new_protected:Nn \__enumext_add_pre_parsep:
1114    {
1115      \int_case:nn { \l__enumext_level_int }
1116        {
1117          { 2 }{
1118              \skip_if_eq:nnF { \l__enumext_parsep_i_skip } { \c_zero_skip }
1119                {
1120                  \skip_add:Nn \l__enumext_multicols_above_ii_skip
1121                    {
1122                      \l__enumext_parsep_i_skip
1123                    }
1124                }
1125            }
1126          { 3 }{
1127              \skip_if_eq:nnF { \l__enumext_parsep_ii_skip } { \c_zero_skip }
1128                {
1129                  \skip_add:Nn \l__enumext_multicols_above_iii_skip
1130                    {
1131                      \l__enumext_parsep_ii_skip
1132                    }
1133                }
1134            }
1135          { 4 }{
1136              \skip_if_eq:nnF { \l__enumext_parsep_iii_skip } { \c_zero_skip }
1137                {
1138                  \skip_add:Nn \l__enumext_multicols_above_iv_skip
1139                    {
1140                      \l__enumext_parsep_iii_skip
1141                    }
1142                }
1143            }
1144        }
1145    }
```

(*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*⟩.

```
1146  \cs_new_protected:Nn \__enumext_multi_addvspace:
1147    {
1148      \__enumext_multi_set_vskip:
1149      \mode_if_vertical:T
1150        {
1151          \skip_add:cn { l__enumext_multicols_above_ \__enumext_level: _skip }
1152            {
1153              \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
1154            }
1155          \skip_add:cn { l__enumext_multicols_below_ \__enumext_level: _skip }
1156            {
1157              \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
1158            }
1159        }
1160      \__enumext_unskip_unkern: % revisar
1161      \par\nopagebreak
1162      \addvspace{ \skip_use:c { l__enumext_multicols_above_ \__enumext_level: _skip } }
1163    }
```

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

12.21.2 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`.

```
1164 \cs_new_protected:Nn \__enumext_keyans_multi_set_vskip:
1165   {
1166     \skip_set:Nn \l__enumext_multicols_above_v_skip
1167       {
1168         \l__enumext_topsep_v_skip
1169       }
1170     \skip_set:Nn \l__enumext_multicols_below_v_skip
1171       {
1172         \l__enumext_topsep_v_skip
1173       }
1174   }
1175 \cs_new_protected:Nn \__enumext_keyans_multi_addvspace:
1176   {
1177     \__enumext_keyans_multi_set_vskip:
1178     \mode_if_vertical:T
1179       {
1180         \skip_add:Nn \l__enumext_multicols_above_v_skip
1181           {
1182             \skip_use:N \l__enumext_partopsep_v_skip
1183           }
1184         \skip_add:Nn \l__enumext_multicols_below_v_skip
1185           {
1186             \skip_use:N \l__enumext_partopsep_v_skip
1187           }
1188       }
1189     \__enumext_unskip_unkern:
1190     \par\nopagebreak
1191     \addvspace{ \l__enumext_multicols_above_v_skip }
1192   }
```

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

12.22 Adjustment of vertical spaces for `minipage`

When nesting a *"list environment"* within the `minipage` environment, the values of the *"vertical spaces"* are lost. Graphically it can be seen like in the figure 8.

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

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

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

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

12.22.1 Adjustment of vertical spaces for `minipage` in `enumext`

__enumext_minipage_set_skip:
__enumext_minipage_add_space:

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

First we will set the value of `\l__enumext_minipage_right_skip` equal to `\topsep`, then we will see if TeX is in ⟨*vertical mode*⟩ and we will add `\partopsep`, followed by that we set the value of `\l__enumext_-minipage_after_skip`.

```
1193 \cs_new_protected:Nn \__enumext_minipage_set_skip:
1194   {
```

```
1195        \skip_set:Nn \l__enumext_minipage_right_skip
1196          {
1197            \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
1198          }
1199        \mode_if_vertical:T
1200          {
1201            \skip_add:Nn \l__enumext_minipage_right_skip
1202              {
1203                \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
1204              }
1205          }
1206        \skip_set_eq:NN \l__enumext_minipage_after_skip \l__enumext_minipage_right_skip
```

We will adjust the values \l__enumext_multicols_above_X_skip and \l__enumext_multicols_-below_X_skip and call the function __enumext_pre_itemsep_skip:.

```
1207        \skip_set_eq:cN
1208          { l__enumext_multicols_above_ \__enumext_level: _skip } \l__enumext_minipage_right_skip
1209        \skip_set_eq:cN
1210          { l__enumext_multicols_below_ \__enumext_level: _skip } \l__enumext_minipage_right_skip
1211        \__enumext_pre_itemsep_skip:
```

If the environment multicols is active, we set \topskip=0pt and then we make \multicolsep have the same value as \l__enumext_multicols_above_X_skip.

```
1212        \int_compare:nNnT
1213          { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } > { 1 }
1214          {
1215            \skip_zero:N \topskip
1216            \skip_set_eq:Nc \multicolsep { l__enumext_multicols_above_ \__enumext_level: _skip }
1217          }
1218      }
```

The function __enumext_minipage_add_space: will apply the spaces on the *"left side"* using \addvspace *"above"* the __enumext_mini_page environment, taking into account whether TeX is in ⟨*horizontal mode*⟩ or ⟨*vertical mode*⟩. Here we use the plain TeX macro \nointerlineskip to prevent baseline *"glue"* being added between the next pair of boxes in a *vertical list*. For the latter we will make some adjustments since the \partopsep parameter comes into play and this affects the *vertical spacing*.

```
1219  \cs_new_protected:Nn \__enumext_minipage_add_space:
1220    {
1221      \__enumext_minipage_set_skip:
1222      \__enumext_unskip_unkern:
1223      \mode_if_vertical:TF
1224        {
1225          \nopagebreak\nointerlineskip
1226        }
1227        {
1228          \par\nopagebreak\nointerlineskip
1229          \skip_zero:c { l__enumext_partopsep_ \__enumext_level: _skip }
1230        }
1231      \int_compare:nNnTF
1232        { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } > { 1 }
1233        {
1234          \addvspace{ 0.445\box_ht:N \strutbox }
1235        }
1236        {
1237          \addvspace{ 0.250\box_ht:N \strutbox }
1238        }
1239    }
```

(*End of definition for* __enumext_minipage_set_skip: *and* __enumext_minipage_add_space:.)

__enumext_pre_itemsep_skip: The function __enumext_pre_itemsep_skip: will adjust the spaces below the environment minipage and the environment multicols if it is nested in it, taking into account the value of \itemsep from the previous level.

```
1240  \cs_new_protected:Nn \__enumext_pre_itemsep_skip:
1241    {
1242      \int_case:nn { \l__enumext_level_int }
1243        {
1244          { 2 }{
1245            \skip_if_eq:nnTF
1246              { \l__enumext_itemsep_i_skip } { \l__enumext_minipage_after_skip }
1247              {
1248                \skip_set:Nn \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
```

```
1249                        \skip_set:Nn \l__enumext_multicols_below_ii_skip { 0.350\box_ht:N \strutbox }
1250                      }
1251                      {
1252                        \dim_compare:nNnT
1253                          { \l__enumext_itemsep_i_skip } < { \l__enumext_minipage_after_skip }
1254                          {
1255                            \skip_sub:Nn
1256                              \l__enumext_minipage_after_skip { \l__enumext_itemsep_i_skip }
1257                            \skip_sub:Nn
1258                              \l__enumext_multicols_below_ii_skip { \l__enumext_itemsep_i_skip }
1259                            \skip_add:Nn
1260                              \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1261                            \skip_add:Nn
1262                              \l__enumext_multicols_below_ii_skip { 0.350\box_ht:N \strutbox }
1263                          }
1264                        \dim_compare:nNnT
1265                          { \l__enumext_itemsep_i_skip } > { \l__enumext_minipage_after_skip }
1266                          {
1267                            \skip_set:Nn \l_tmpa_skip
1268                              {
1269                                \l__enumext_itemsep_i_skip - \l__enumext_minipage_after_skip
1270                              }
1271                            \skip_sub:Nn
1272                              \l__enumext_minipage_after_skip { \l__enumext_itemsep_i_skip }
1273                            \skip_sub:Nn
1274                              \l__enumext_multicols_below_ii_skip { \l__enumext_itemsep_i_skip }
1275                            \skip_add:Nn
1276                              \l__enumext_minipage_after_skip
1277                              { 0.150\box_ht:N \strutbox + \l_tmpa_skip }
1278                            \skip_add:Nn
1279                              \l__enumext_multicols_below_ii_skip
1280                              { 0.350\box_ht:N \strutbox + \l_tmpa_skip }
1281                          }
1282                      }
1283                  }
1284          { 3 }{
1285                  \skip_if_eq:nnTF
1286                    { \l__enumext_itemsep_ii_skip } { \c_zero_skip }
1287                    {
1288                      \skip_set:Nn \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1289                      \skip_set:Nn \l__enumext_multicols_below_iii_skip { 0.350\box_ht:N \strutbox }
1290                    }
1291                    {
1292                        \dim_compare:nNnT
1293                          { \l__enumext_itemsep_ii_skip } < { \l__enumext_minipage_after_skip }
1294                          {
1295                            \skip_sub:Nn
1296                              \l__enumext_minipage_after_skip { \l__enumext_itemsep_ii_skip }
1297                            \skip_sub:Nn
1298                              \l__enumext_multicols_below_iii_skip { \l__enumext_itemsep_ii_skip }
1299                            \skip_add:Nn
1300                              \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1301                            \skip_add:Nn
1302                              \l__enumext_multicols_below_iii_skip { 0.350\box_ht:N \strutbox }
1303                          }
1304                        \dim_compare:nNnT
1305                          { \l__enumext_itemsep_ii_skip } > { \l__enumext_minipage_after_skip }
1306                          {
1307                            \skip_set:Nn \l_tmpa_skip
1308                              {
1309                                \l__enumext_itemsep_ii_skip - \l__enumext_minipage_after_skip
1310                              }
1311                            \skip_sub:Nn
1312                              \l__enumext_minipage_after_skip { \l__enumext_itemsep_ii_skip }
1313                            \skip_sub:Nn
1314                              \l__enumext_multicols_below_iii_skip { \l__enumext_itemsep_ii_skip }
1315                            \skip_add:Nn
1316                              \l__enumext_minipage_after_skip
1317                              { 0.150\box_ht:N \strutbox + \l_tmpa_skip }
1318                            \skip_add:Nn
1319                              \l__enumext_multicols_below_iii_skip
```

```
1320                              { 0.350\box_ht:N \strutbox + \l_tmpa_skip }
1321                            }
1322                          }
1323                        }
1324              { 4 }{
1325                  \skip_if_eq:nnTF { \l__enumext_itemsep_iii_skip } { \c_zero_skip }
1326                    {
1327                      \skip_set:Nn \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1328                      \skip_set:Nn \l__enumext_multicols_below_iv_skip { 0.350\box_ht:N \strutbox }
1329                    }
1330                    {
1331                      \dim_compare:nNnT
1332                        { \l__enumext_itemsep_iii_skip } < { \l__enumext_minipage_after_skip }
1333                        {
1334                          \skip_sub:Nn
1335                            \l__enumext_minipage_after_skip { \l__enumext_itemsep_iii_skip }
1336                          \skip_sub:Nn
1337                            \l__enumext_multicols_below_iv_skip { \l__enumext_itemsep_iii_skip }
1338                          \skip_add:Nn
1339                            \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1340                          \skip_add:Nn
1341                            \l__enumext_multicols_below_iv_skip { 0.350\box_ht:N \strutbox }
1342                        }
1343                      \dim_compare:nNnT
1344                        { \l__enumext_itemsep_iii_skip } > { \l__enumext_minipage_after_skip }
1345                        {
1346                          \skip_set:Nn \l_tmpa_skip
1347                            {
1348                              \l__enumext_itemsep_iii_skip - \l__enumext_minipage_after_skip
1349                            }
1350                          \skip_sub:Nn
1351                            \l__enumext_minipage_after_skip { \l__enumext_itemsep_iii_skip }
1352                          \skip_sub:Nn
1353                            \l__enumext_multicols_below_iv_skip { \l__enumext_itemsep_iii_skip }
1354                          \skip_add:Nn
1355                            \l__enumext_minipage_after_skip
1356                            { 0.150\box_ht:N \strutbox + \l_tmpa_skip }
1357                          \skip_add:Nn
1358                            \l__enumext_multicols_below_iv_skip
1359                            { 0.350\box_ht:N \strutbox + \l_tmpa_skip }
1360                        }
1361                    }
1362                }
1363            }
1364        }
```

(*End of definition for* __enumext_pre_itemsep_skip:.)

12.22.2 Adjustment of vertical spaces for `minipage` in keyans

__enumext_keyans_minipage_set_skip:
__enumext_keyans_minipage_add_space:
__enumext_keyans_pre_itemsep_skip:

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_page environment in keyans. The implementation of this function is the same as the one used in enumext.

```
1365  \cs_new_protected:Nn \__enumext_keyans_minipage_set_skip:
1366    {
1367      \skip_zero:N \l__enumext_minipage_after_skip
1368      \skip_zero:N \l__enumext_minipage_left_skip
1369      \skip_zero:N \l__enumext_minipage_right_skip
1370      \skip_set:Nn \l__enumext_minipage_right_skip
1371        {
1372          \l__enumext_topsep_v_skip
1373        }
1374      \mode_if_vertical:T
1375        {
1376          \skip_add:Nn \l__enumext_minipage_right_skip
1377            {
1378              \l__enumext_partopsep_v_skip
1379            }
1380        }
1381      \skip_set_eq:NN \l__enumext_minipage_after_skip \l__enumext_minipage_right_skip
1382      \skip_set_eq:NN \l__enumext_multicols_above_v_skip \l__enumext_minipage_right_skip
1383      \skip_set_eq:NN \l__enumext_multicols_below_v_skip \l__enumext_minipage_right_skip
```

```
1384        \__enumext_keyans_pre_itemsep_skip:
1385        \int_compare:nNnT { \l__enumext_columns_v_int } > { 1 }
1386          {
1387            \skip_zero:N \topskip
1388            \skip_set_eq:NN \multicolsep \l__enumext_minipage_right_skip
1389          }
1390      }
1391  \cs_new_protected:Nn \__enumext_keyans_minipage_add_space:
1392    {
1393      \__enumext_keyans_minipage_set_skip:
1394      \__enumext_unskip_unkern:
1395      \mode_if_vertical:TF
1396        {
1397            \nopagebreak\nointerlineskip
1398        }
1399        {
1400            \par\nopagebreak\nointerlineskip
1401            \skip_zero:N \l__enumext_partopsep_v_skip
1402        }
1403      \int_compare:nNnTF { \l__enumext_columns_v_int } > { 1 }
1404        {
1405            \addvspace{ 0.445\box_ht:N \strutbox }
1406        }
1407        {
1408            \addvspace{ 0.250\box_ht:N \strutbox }
1409        }
1410      }
1411  \cs_new_protected:Nn \__enumext_keyans_pre_itemsep_skip:
1412    {
1413      \skip_if_eq:nnTF
1414        { \l__enumext_itemsep_i_skip } { \l__enumext_minipage_after_skip }
1415        {
1416            \skip_set:Nn \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1417            \skip_set:Nn \l__enumext_multicols_below_v_skip { 0.350\box_ht:N \strutbox }
1418        }
1419        {
1420            \dim_compare:nNnT
1421              { \l__enumext_itemsep_i_skip } < { \l__enumext_minipage_after_skip }
1422              {
1423                \skip_sub:Nn \l__enumext_minipage_after_skip { \l__enumext_itemsep_i_skip }
1424                \skip_sub:Nn \l__enumext_multicols_below_v_skip { \l__enumext_itemsep_i_skip }
1425                \skip_add:Nn \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1426                \skip_add:Nn \l__enumext_multicols_below_v_skip { 0.350\box_ht:N \strutbox }
1427              }
1428            \dim_compare:nNnT
1429              { \l__enumext_itemsep_i_skip } > { \l__enumext_minipage_after_skip }
1430              {
1431                \skip_set:Nn \l_tmpa_skip
1432                  {
1433                    \l__enumext_itemsep_i_skip - \l__enumext_minipage_after_skip
1434                  }
1435                \skip_sub:Nn \l__enumext_minipage_after_skip { \l__enumext_itemsep_i_skip }
1436                \skip_sub:Nn \l__enumext_multicols_below_v_skip { \l__enumext_itemsep_i_skip }
1437                \skip_add:Nn \l__enumext_minipage_after_skip
1438                  { 0.150\box_ht:N \strutbox + \l_tmpa_skip }
1439                \skip_add:Nn \l__enumext_multicols_below_v_skip
1440                  { 0.350\box_ht:N \strutbox + \l_tmpa_skip }
1441              }
1442        }
1443    }
```

(*End of definition for* __enumext_keyans_minipage_set_skip:, __enumext_keyans_minipage_add_space:, *and* __enumext_keyans_pre_itemsep_skip:.)

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

`\__enumext_mini_set_vskip_vii:`
`\__enumext_mini_set_vskip_viii:`

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

```
1444  \cs_new_protected:Nn \__enumext_mini_set_vskip_vii:
1445    {
1446      \skip_zero_new:N \l__enumext_minipage_left_skip
```

```
1447    \skip_gzero_new:N \g__enumext_minipage_right_skip
1448    \skip_gzero_new:N \g__enumext_minipage_after_skip
1449    \skip_if_eq:nnTF { \l__enumext_topsep_vii_skip } { \c_zero_skip }
1450      {
1451        \skip_set:Nn \l__enumext_minipage_left_skip { 0.5\box_dp:N \strutbox }
1452        \skip_gset:Nn \g__enumext_minipage_right_skip { 0.325\box_dp:N \strutbox }
1453      }
1454      {
1455        \skip_set:Nn \l__enumext_minipage_left_skip { 0.5875\box_dp:N \strutbox }
1456        \skip_gset:Nn \g__enumext_minipage_right_skip
1457          {
1458            \l__enumext_topsep_vii_skip
1459          }
1460        \skip_gset:Nn \g__enumext_minipage_after_skip
1461          {
1462            0.325\box_dp:N \strutbox + \l__enumext_topsep_vii_skip
1463          }
1464      }
1465    }
1466  \cs_new_protected:Nn \__enumext_mini_set_vskip_viii:
1467    {
1468      \skip_zero_new:N \l__enumext_minipage_after_skip
1469      \skip_zero_new:N \l__enumext_minipage_left_skip
1470      \skip_zero_new:N \l__enumext_minipage_right_skip
1471      \skip_if_eq:nnTF { \l__enumext_topsep_viii_skip } { \c_zero_skip }
1472        {
1473          \skip_set:Nn \l__enumext_minipage_left_skip
1474            {
1475              0.5\box_dp:N \strutbox
1476            }
1477          \skip_set:Nn \l__enumext_minipage_right_skip
1478            {
1479              \l__enumext_partopsep_viii_skip
1480            }
1481          \skip_set:Nn \l__enumext_minipage_after_skip
1482            {
1483              1.6\box_dp:N \strutbox
1484            }
1485        }
1486        {
1487          \skip_set:Nn \l__enumext_minipage_left_skip
1488            {
1489              0.5875\box_dp:N \strutbox
1490            }
1491          \skip_set:Nn \l__enumext_minipage_right_skip
1492            {
1493              \l__enumext_topsep_viii_skip
1494            }
1495          \skip_set:Nn \l__enumext_minipage_after_skip
1496            {
1497              0.325\box_dp:N \strutbox + \l__enumext_topsep_viii_skip
1498            }
1499        }
1500    }
```

(*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_page 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.

```
1501  \cs_new_protected:Nn \__enumext_mini_addvspace_vii:
1502    {
1503      \__enumext_mini_set_vskip_vii:
1504      \par\nopagebreak
1505      \addvspace { \l__enumext_minipage_left_skip }
1506    }
1507  \cs_new_protected:Nn \__enumext_mini_addvspace_viii:
1508    {
1509      \__enumext_mini_set_vskip_viii:
```

```
1510       \par\nopagebreak
1511       \addvspace { \l__enumext_minipage_left_skip }
1512     }
```

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

12.22.4 The command \miniright

The command \miniright will close the __enumext_mini_page environment on the *"left side"*, open the __enumext_mini_page 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_page on the *"right side"*.

\miniright First we will perform some checks to prevent the command from being executed outside the enumext environment or somewhere inappropriate then we will call the internal functions to execute it in the enumext and keyans environments.

```
1513 \NewDocumentCommand \miniright { s }
1514   {
1515     \int_compare:nNnT { \l__enumext_keyans_pic_level_int } = { 1 }
1516       {
1517         \msg_error:nnn { enumext } { wrong-miniright-place }
1518       }
1519     % outside
1520     \bool_lazy_and:nnT
1521       { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
1522       { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
1523       {
1524         \msg_error:nnn { enumext } { wrong-miniright-place }
1525       }
1526     % starred env
1527     \bool_if:NT \l__enumext_starred_bool
1528       {
1529         \msg_error:nnn { enumext } { wrong-miniright-starred }
1530       }
1531     \int_compare:nNnTF { \l__enumext_keyans_level_int } = { 1 }
1532       {
1533         \__enumext_keyans_mini_right_cmd:n {#1}
1534       }
1535       { \__enumext_mini_right_cmd:n {#1} }
1536   }
```

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

__enumext_mini_right_cmd:n The function __enumext_mini_right_cmd:n takes as argument the *starred* '*' of the \miniright command in the enumext environment. We check if the mini-env key is active via the variable \l__enumext_-minipage_right_X_dim, if so we close the multicols environment with the __enumext_mini_page environment on the *"left side"*, then we open the __enumext_mini_page 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.

```
1537 \cs_new_protected:Npn \__enumext_mini_right_cmd:n #1
1538   {
1539     \dim_compare:nNnTF
1540       { \dim_use:c { l__enumext_minipage_right_ \__enumext_level: _dim } } > { \c_zero_dim }
1541       {
1542         \__enumext_multicols_stop:
1543         \int_compare:nNnT
1544           { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } = { 1 }
1545           {
1546             \par\addvspace{ \l__enumext_minipage_after_skip }
1547           }
1548         \end__enumext_mini_page
1549         \hfill
1550         \__enumext_mini_page{ \dim_use:c { l__enumext_minipage_right_ \__enumext_level: _dim } }
1551           \par\nointerlineskip
1552           \addvspace { \l__enumext_minipage_right_skip }
1553           \bool_if:nF {#1}
1554             {
1555               \centering
1556             }
1557           \int_gzero:N \g__enumext_minipage_stat_int
```

```
1558        }
1559      { \msg_error:nnn { enumext } { wrong-miniright-use } }
1560    % paranoia
1561    \RenewDocumentCommand \miniright { s }
1562      {
1563        \msg_error:nn { enumext } { many-miniright-used }
1564      }
1565  }
```

(*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.

```
1566  \cs_new_protected:Npn \__enumext_keyans_mini_right_cmd:n #1
1567    {
1568      \dim_compare:nNnTF { \l__enumext_minipage_right_v_dim } > { \c_zero_dim }
1569        {
1570          \__enumext_keyans_multicols_stop:
1571          \int_compare:nNnT { \l__enumext_columns_v_int } = { 1 }
1572            {
1573              \par\addvspace{ \l__enumext_minipage_after_skip }
1574            }
1575          \end__enumext_mini_page
1576          \hfill
1577          \__enumext_mini_page{ \l__enumext_minipage_right_v_dim }
1578            \par\nointerlineskip
1579            \addvspace { \l__enumext_minipage_right_skip }
1580            \bool_if:nF {#1}
1581              {
1582                \centering
1583              }
1584            \int_gzero:N \g__enumext_minipage_stat_int
1585        }
1586      { \msg_error:nnn { enumext } { wrong-miniright-use } }
1587    % paranoia
1588    \RenewDocumentCommand \miniright { s }
1589      {
1590        \msg_error:nn { enumext } { many-miniright-used }
1591      }
1592  }
```

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

12.23   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.

```
1593  \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1594    {
1595      \keys_define:nn { enumext / #1 }
1596        {
1597          above  .skip_set:c = { l__enumext_vspace_above_#2_skip },
1598          above  .value_required:n = true,
1599          above* .code:n     = \bool_set_true:c { l__enumext_vspace_a_star_#2_bool }
1600                               \keys_set:nn { enumext / #1 } { above = {##1} },
1601          above* .value_required:n = true,
1602          below  .skip_set:c = { l__enumext_vspace_below_#2_skip },
1603          below  .value_required:n = true,
1604          below* .code:n     = \bool_set_true:c { l__enumext_vspace_b_star_#2_bool }
1605                               \keys_set:nn { enumext / #1 } { below = {##1} },
1606          below* .value_required:n = true,
1607        }
1608    }
1609  \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }
```

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

12.23.1 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.

```
1610  \cs_new_protected:Nn \__enumext_vspace_above:
1611    {
1612      \skip_if_eq:nnF
1613        { \skip_use:c { l__enumext_vspace_above_ \__enumext_level: _skip } } { \c_zero_skip }
1614        {
1615          \bool_if:cTF { l__enumext_vspace_a_star_ \__enumext_level: _bool }
1616            {
1617              \vspace*{ \skip_use:c { l__enumext_vspace_above_ \__enumext_level: _skip } }
1618            }
1619            {
1620              \vspace { \skip_use:c { l__enumext_vspace_above_ \__enumext_level: _skip } }
1621            }
1622        }
1623    }
```

(*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.

```
1624  \cs_new_protected:Nn \__enumext_vspace_below:
1625    {
1626      \skip_if_eq:nnF
1627        { \skip_use:c { l__enumext_vspace_below_ \__enumext_level: _skip } } { \c_zero_skip }
1628        {
1629          \bool_if:cTF { l__enumext_vspace_b_star_ \__enumext_level: _bool }
1630            {
1631              \vspace*{ \skip_use:c { l__enumext_vspace_below_ \__enumext_level: _skip } }
1632            }
1633            {
1634              \vspace { \skip_use:c { l__enumext_vspace_below_ \__enumext_level: _skip } }
1635            }
1636        }
1637    }
```

(*End of definition for* __enumext_vspace_below:.)

12.23.2 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.

```
1638  \cs_new_protected:Nn \__enumext_vspace_above_v:
1639    {
1640      \skip_if_eq:nnF { \l__enumext_vspace_above_v_skip } { \c_zero_skip }
1641        {
1642          \bool_if:NTF \l__enumext_vspace_a_star_v_bool
1643            {
1644              \vspace*{ \l__enumext_vspace_above_v_skip }
1645            }
1646            { \vspace { \l__enumext_vspace_above_v_skip } }
1647        }
1648    }
```

(*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.

```
1649  \cs_new_protected:Nn \__enumext_vspace_below_v:
1650    {
1651      \skip_if_eq:nnF { \l__enumext_vspace_below_v_skip } { \c_zero_skip }
1652        {
1653          \bool_if:NTF \l__enumext_vspace_b_star_v_bool
1654            {
1655              \vspace*{ \l__enumext_vspace_below_v_skip }
1656            }
1657            { \vspace { \l__enumext_vspace_below_v_skip } }
1658        }
1659    }
```

(*End of definition for* __enumext_vspace_below_v:.)

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

__enumext_vspace_above_vii:
__enumext_vspace_above_viii:

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

```
1660 \cs_new_protected:Nn \__enumext_vspace_above_vii:
1661   {
1662     \skip_if_eq:nnF { \l__enumext_vspace_above_vii_skip } { \c_zero_skip }
1663       {
1664         \bool_if:NTF \l__enumext_vspace_a_star_vii_bool
1665           {
1666             \vspace*{ \l__enumext_vspace_above_vii_skip }
1667           }
1668           { \vspace { \l__enumext_vspace_above_vii_skip } }
1669       }
1670   }
1671 \cs_new_protected:Nn \__enumext_vspace_above_viii:
1672   {
1673     \skip_if_eq:nnF { \l__enumext_vspace_above_viii_skip } { \c_zero_skip }
1674       {
1675         \bool_if:NTF \l__enumext_vspace_a_star_viii_bool
1676           {
1677             \vspace*{ \l__enumext_vspace_above_viii_skip }
1678           }
1679           { \vspace { \l__enumext_vspace_above_viii_skip } }
1680       }
1681   }
```

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

__enumext_vspace_below_vii:
__enumext_vspace_below_viii:

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

```
1682 \cs_new_protected:Nn \__enumext_vspace_below_vii:
1683   {
1684     \skip_if_eq:nnF { \l__enumext_vspace_below_vii_skip } { \c_zero_skip }
1685       {
1686         \bool_if:NTF \l__enumext_vspace_b_star_vii_bool
1687           {
1688             \vspace*{ \l__enumext_vspace_below_vii_skip }
1689           }
1690           { \vspace { \l__enumext_vspace_below_vii_skip } }
1691       }
1692   }
1693 \cs_new_protected:Nn \__enumext_vspace_below_viii:
1694   {
1695     \skip_if_eq:nnF { \l__enumext_vspace_below_viii_skip } { \c_zero_skip }
1696       {
1697         \bool_if:NTF \l__enumext_vspace_b_star_viii_bool
1698           {
1699             \vspace*{ \l__enumext_vspace_below_viii_skip }
1700           }
1701           { \vspace { \l__enumext_vspace_below_viii_skip } }
1702       }
1703   }
```

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

12.24 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*.

```
1704 \cs_set_protected:Npn \__enumext_tmp:n #1
1705   {
1706     \keys_define:nn { enumext / #1 }
1707       {
1708         series  .str_set:N = \l__enumext_series_str,
1709         series  .value_required:n = true,
1710         resume  .code:n = \__enumext_resume_series:n {##1},
1711         resume* .code:n = \__enumext_resume_starred:,
```

```
1712          resume* .value_forbidden:n = true,
1713        }
1714    }
1715  \clist_map_inline:nn { level-1, enumext* } { \__enumext_tmp:n {#1} }
```

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

12.24.1 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.

```
1716  \cs_new:Npn \__enumext_filter_series:n #1
1717    {
1718      \use:e
1719        {
1720          \keyval_parse:NNn
1721            \__enumext_filter_series_key:n
1722            \__enumext_filter_series_pair:nn {#1}
1723        }
1724    }
```

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

```
1725  \cs_new:Npn \__enumext_filter_series_key:n #1
1726    {
1727      \str_case:nnF {#1}
1728        {
1729          { resume } {} { resume* } {} { base-fix } {}
1730        }
1731        { , { \exp_not:n {#1} } }
1732    }
```

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, start*, save-ans and save-key keys.

```
1733  \cs_new:Npn \__enumext_filter_series_pair:nn #1#2
1734    {
1735      \str_case:nnF {#1}
1736        {
1737          { series } {} { resume } {} { start } {}
1738          { start* } {}  { save-ans } {} { save-key } {}
1739        }
1740        { , { \exp_not:n {#1} } = { \exp_not:n {#2} } }
1741    }
```

(*End of definition for* __enumext_filter_series:n*,* __enumext_filter_series_key:n*, and* __enumext_filter_series_-
pair:nn*.*)

__enumext_parse_series:n
__enumext_resume_last:n

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

🎯 The value of boolean variable \l_enumext_resume_active_bool is set to true by the function __enumext_resume_-counter:n which is used by the keys resume and resume*, in this case we must Make sure it is set to false so that it does not overwrite the default filtered ⟨*keys*⟩. This function is passed to the function __enumext_parse_keys:n in the enumext environment definition (§12.38) and to the function __enumext_parse_keys_vii:n in the enumext* environment definition (§12.43).

```
1742  \cs_new_protected:Npn \__enumext_parse_series:n #1
1743    {
1744      \str_if_empty:NTF \l__enumext_series_str
1745        {
1746          \bool_if:NF \l__enumext_resume_active_bool
1747            {
1748              \__enumext_resume_last:n {#1}
1749            }
1750        }
1751        {
1752          \tl_gclear_new:c { g__enumext_series_ \l__enumext_series_str _tl }
1753          \tl_gset:ce { g__enumext_series_ \l__enumext_series_str _tl }
1754            { \__enumext_filter_series:n {#1} }
1755          \int_if_exist:cF { g__enumext_series_ \l__enumext_series_str _int }
1756            {
```

```
1757              \int_new:c { g__enumext_series_ \l__enumext_series_str _int }
1758            }
1759          }
1760        }
```

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.

```
1761  \cs_new_protected:Npn \__enumext_resume_last:n #1
1762    {
1763      \bool_if:NT \l__enumext_standar_first_bool
1764        {
1765          \tl_gclear:N \g__enumext_standar_series_tl
1766          \tl_gset:Ne \g__enumext_standar_series_tl { \__enumext_filter_series:n {#1} }
1767        }
1768      \bool_if:NT \l__enumext_starred_first_bool
1769        {
1770          \tl_gclear:N \g__enumext_starred_series_tl
1771          \tl_gset:Ne \g__enumext_starred_series_tl { \__enumext_filter_series:n {#1} }
1772        }
1773    }
```

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

12.24.2   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 (§12.38) and the enumext* environment definition (§12.43).

```
1774  \cs_new_protected:Nn \__enumext_resume_save_counter:
1775    {
1776      \bool_if:NT \g__enumext_standar_bool
1777        {
1778          \tl_if_empty:NF \l__enumext_series_str
1779            {
1780              \int_gset_eq:cN
1781                { g__enumext_series_ \l__enumext_series_str _int } \value{enumXi}
1782            }
1783          \tl_if_empty:NTF \l__enumext_resume_name_tl
1784            {
1785              \str_if_empty:NT \l__enumext_series_str
1786                {
1787                  \int_gset_eq:NN \g__enumext_resume_int \value{enumXi}
1788                }
1789            }
1790            {
1791              \int_if_exist:cT { g__enumext_series_ \l__enumext_resume_name_tl _int }
1792                {
1793                  \int_gset_eq:cN
1794                    { g__enumext_series_ \l__enumext_resume_name_tl _int } \value{enumXi}
1795                }
1796            }
1797          \int_if_exist:cT { g__enumext_resume_ \l__enumext_store_name_tl _int }
1798            {
1799              \int_gset_eq:cN
1800                { g__enumext_resume_ \l__enumext_store_name_tl _int } \value{enumXi}
1801            }
1802        }
1803      \bool_if:NT \g__enumext_starred_bool
1804        {
1805          \tl_if_empty:NF \l__enumext_series_str
1806            {
```

```
1807                      \int_gset_eq:cN
1808                        { g__enumext_series_ \l__enumext_series_str _int } \value{enumXvii}
1809                    }
1810                \tl_if_empty:NTF \l__enumext_resume_name_tl
1811                  {
1812                    \str_if_empty:NT \l__enumext_series_str
1813                      {
1814                        \int_gset_eq:NN \g__enumext_resume_vii_int \value{enumXvii}
1815                      }
1816                  }
1817                  {
1818                    \int_if_exist:cT { g__enumext_series_ \l__enumext_resume_name_tl _int }
1819                      {
1820                        \int_gset_eq:cN
1821                          { g__enumext_series_ \l__enumext_resume_name_tl _int } \value{enumXvii}
1822                      }
1823                  }
1824                \int_if_exist:cT { g__enumext_resume_ \l__enumext_store_name_tl _int }
1825                  {
1826                    \int_gset_eq:cN
1827                      { g__enumext_resume_ \l__enumext_store_name_tl _int } \value{enumXvii}
1828                  }
1829              }
1830          }
```

(*End of definition for* __enumext_resume_save_counter:.)

12.24.3   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.

```
1831  \cs_new_protected:Npn \__enumext_resume_series:n #1
1832    {
1833      \tl_if_empty:nTF {#1}
1834        {
1835          \__enumext_resume_counter:n { }
1836        }
1837        {
1838          \tl_if_exist:cTF { g__enumext_series_ \tl_to_str:n {#1} _tl }
1839            {
1840              \__enumext_resume_counter:n {#1}
1841              \bool_if:NT \g__enumext_standar_bool
1842                {
1843                  \keys_set:nv { enumext / level-1 }
1844                    { g__enumext_series_ \tl_to_str:n {#1} _tl }
1845                }
1846              \bool_if:NT \g__enumext_starred_bool
1847                {
1848                  \keys_set:nv { enumext / enumext* }
1849                    { g__enumext_series_ \tl_to_str:n {#1} _tl }
1850                }
1851            }
1852            {
1853              \bool_if:NT \g__enumext_standar_bool
1854                {
1855                  \msg_error:nnn { enumext } { unknown-series } {#1}
1856                }
1857              \bool_if:NT \g__enumext_starred_bool
1858                {
1859                  \msg_error:nnn { enumext } { unknown-series } {#1}
1860                }
1861            }
1862        }
1863    }
```

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

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.

```
1864  \cs_new_protected:Npn \__enumext_resume_counter:n #1
1865    {
1866      \bool_set_true:N \l__enumext_resume_active_bool
1867      \tl_set:Nn \l__enumext_resume_name_tl {#1}
1868      \tl_if_empty:NTF \l__enumext_resume_name_tl
1869        {
1870          \__enumext_resume_counter:
1871        }
1872        {
1873          \__enumext_resume_counter_series:
1874        }
1875      \__enumext_resume_counter_save_ans:
1876    }
```

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.

```
1877  \cs_new_protected:Nn \__enumext_resume_counter:
1878    {
1879      \bool_if:NT \g__enumext_standar_bool
1880        {
1881          \int_gincr:N \g__enumext_resume_int
1882          \int_set_eq:NN \l__enumext_start_i_int \g__enumext_resume_int
1883        }
1884      \bool_if:NT \g__enumext_starred_bool
1885        {
1886          \int_gincr:N \g__enumext_resume_vii_int
1887          \int_set_eq:NN \l__enumext_start_vii_int \g__enumext_resume_vii_int
1888        }
1889    }
```

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.

```
1890  \cs_new_protected:Nn \__enumext_resume_counter_series:
1891    {
1892      \bool_if:NT \g__enumext_standar_bool
1893        {
1894          \int_set:Nn \l__enumext_start_i_int
1895            {
1896              \int_use:c { g__enumext_series_ \l__enumext_resume_name_tl _int } + 1
1897            }
1898        }
1899      \bool_if:NT \g__enumext_starred_bool
1900        {
1901          \int_set:Nn \l__enumext_start_vii_int
1902            {
1903              \int_use:c { g__enumext_series_ \l__enumext_resume_name_tl _int } + 1
1904            }
1905        }
1906    }
```

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.

```
1907  \cs_new_protected:Nn \__enumext_resume_counter_save_ans:
1908    {
1909      \bool_lazy_and:nnT
1910        { \bool_if_p:N \l__enumext_standar_first_bool }
1911        { \bool_if_p:N \l__enumext_store_active_bool }
1912        {
1913          \int_set:Nn \l__enumext_start_i_int
1914            {
1915              \int_use:c { g__enumext_resume_ \l__enumext_store_name_tl _int } + 1
1916            }
```

```
1917          }
1918      \bool_lazy_and:nnT
1919        { \bool_if_p:N \l__enumext_starred_first_bool }
1920        { \bool_if_p:N \l__enumext_store_active_bool }
1921        {
1922          \int_set:Nn \l__enumext_start_vii_int
1923            {
1924              \int_use:c { g__enumext_resume_ \l__enumext_store_name_tl _int } + 1
1925            }
1926        }
1927      }
```

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

12.24.4   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.

```
1928  \cs_new_protected:Nn \__enumext_resume_starred:
1929    {
1930      \bool_if:NT \g__enumext_standar_bool
1931        {
1932          \tl_if_empty:NF \g__enumext_standar_series_tl
1933            {
1934              \__enumext_resume_counter:n { }
1935              \keys_set:nV { enumext / level-1 } \g__enumext_standar_series_tl
1936            }
1937        }
1938      \bool_if:NT \g__enumext_starred_bool
1939        {
1940          \tl_if_empty:NF \g__enumext_starred_series_tl
1941            {
1942              \__enumext_resume_counter:n { }
1943              \keys_set:nV { enumext / enumext* } \g__enumext_starred_series_tl
1944            }
1945        }
1946    }
```

(*End of definition for* __enumext_resume_starred:.)

12.25   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.

12.25.1   Setting save-ans key

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

```
1947  \cs_set_protected:Npn \__enumext_tmp:n #1
1948    {
1949      \keys_define:nn { enumext / #1 }
1950        {
1951          save-ans .code:n = \__enumext_storing_set:n {##1},
1952          save-ans .value_required:n = true,
1953        }
1954    }
1955  \clist_map_inline:nn { level-1, enumext* } { \__enumext_tmp:n {#1} }
```

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

12.25.2   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 display
__enumext_stop_save_ans_msg: 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:.

```
1956  \cs_new_protected:Nn \__enumext_start_save_ans_msg:
1957    {
1958      \msg_term:nnVV { enumext } { save-ans-log }
1959        \g__enumext_envir_name_tl \l__enumext_store_name_tl
1960    }
```

```
1961 \cs_new_protected:Nn \__enumext_stop_save_ans_msg:
1962   {
1963     \msg_term:nnVV { enumext } { save-ans-log-hook }
1964       \g__enumext_envir_name_tl \g__enumext_store_name_tl
1965   }
```

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

__enumext_storing_set:n
__enumext_storing_exec:

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

```
1966 \cs_new_protected:Npn \__enumext_storing_set:n #1
1967   {
1968     \tl_set:Ne \l__enumext_store_name_tl {#1}
1969     \tl_if_empty:NTF \l__enumext_store_name_tl
1970       {
1971         \bool_lazy_or:nnT
1972           { \l__enumext_standar_first_bool } { \l__enumext_starred_first_bool }
1973           {
1974             \msg_error:nnV { enumext } { save-ans-empty } \g__enumext_envir_name_tl
1975           }
1976       }
1977       {
1978         \bool_lazy_or:nnT
1979           { \l__enumext_standar_first_bool } { \l__enumext_starred_first_bool }
1980           {
1981             \__enumext_start_save_ans_msg:
1982             \__enumext_storing_exec:
1983           }
1984       }
1985   }
```

The function __enumext_storing_exec: will set to true the variable \l__enumext_store_active_bool which activates the use of the \anskey command and the keyans, keyans* and keyanspic environments and will set to true the variable \l__enumext_check_answers_bool used for checking answers by the check-ans and no-store keys, copy {⟨*store name*⟩} into the global variable \g__enumext_store_name_tl and execute the function __enumext_anskey_env_make:V creating the environment anskey* (§12.30). 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*.

```
1986 \cs_new_protected:Nn \__enumext_storing_exec:
1987   {
1988     \bool_set_true:N \l__enumext_store_active_bool
1989     \bool_set_true:N \l__enumext_check_answers_bool
1990     \tl_gset:NV \g__enumext_store_name_tl \l__enumext_store_name_tl
1991     \__enumext_anskey_env_make:V \l__enumext_store_name_tl
1992     \prop_if_exist:cF { g__enumext_ \l__enumext_store_name_tl _prop }
1993       {
1994         \msg_log:nnV { enumext } { store-prop } \l__enumext_store_name_tl
1995         \prop_new:c { g__enumext_ \l__enumext_store_name_tl _prop }
1996       }
1997     \seq_if_exist:cF { g__enumext_ \l__enumext_store_name_tl _seq }
1998       {
1999         \msg_log:nnV { enumext } { store-seq } \l__enumext_store_name_tl
2000         \seq_new:c { g__enumext_ \l__enumext_store_name_tl _seq }
2001       }
2002     \int_if_exist:cF { g__enumext_resume_ \l__enumext_store_name_tl _int }
2003       {
2004         \msg_log:nnV { enumext } { store-int } \l__enumext_store_name_tl
2005         \int_new:c { g__enumext_resume_ \l__enumext_store_name_tl _int }
2006       }
2007   }
```

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

12.25.3   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.

12.25.4 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.

```
2008 \cs_set_protected:Npn \__enumext_tmp:n #1
2009   {
2010     \keys_define:nn { enumext / #1 }
2011       {
2012         check-ans .bool_set:N = \l__enumext_check_ans_key_bool,
2013         check-ans .initial:n  = false,
2014         check-ans .value_required:n = true,
2015         no-store  .code:n = {
2016                             \bool_set_false:N \l__enumext_check_answers_bool
2017                             \bool_set_false:N \l__enumext_check_ans_key_bool
2018                            },
2019         no-store  .value_forbidden:n = true,
2020       }
2021   }
2022 \clist_map_inline:nn
2023   {
2024     level-1, level-2, level-3, level-4, enumext*
2025   }
2026   { \__enumext_tmp:n {#1} }
```

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

12.25.5 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.

```
2027 \cs_new_protected:Nn \__enumext_check_ans_active:
2028   {
2029     \tl_if_empty:NF \l__enumext_store_name_tl
2030       {
2031         \bool_if:NT \l__enumext_check_answers_bool
2032           {
2033             \__enumext_check_ans_level:
2034           }
2035       }
2036   }
```

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

```
2037 \cs_new_protected:Nn \__enumext_check_ans_level:
2038   {
2039     \int_case:nn { \l__enumext_level_int }
2040       {
2041         { 1 }{
2042               \bool_lazy_all:nT
2043                 {
2044                   { \bool_if_p:N \g__enumext_starred_bool }
2045                   { \int_compare_p:nNn { \l__enumext_level_h_int } = { 1 } }
2046                 }
2047                 {
2048                   \int_gdecr:N \g__enumext_item_number_int
2049                   \bool_set_false:N \l__enumext_item_number_bool
2050                 }
2051             }
2052         { 2 }{
2053               \int_gdecr:N \g__enumext_item_number_int
2054               \bool_set_false:N \l__enumext_item_number_bool
2055             }
2056         { 3 }{
2057               \int_gdecr:N \g__enumext_item_number_int
2058               \bool_set_false:N \l__enumext_item_number_bool
2059             }
2060         { 4 }{
2061               \int_gdecr:N \g__enumext_item_number_int
2062               \bool_set_false:N \l__enumext_item_number_bool
2063             }
2064       }
```

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.

```
2065     \int_case:nn { \l__enumext_level_h_int }
2066       {
2067         { 1 }{
2068               \bool_lazy_all:nT
2069                 {
2070                   { \bool_if_p:N \g__enumext_standar_bool }
2071                   { \int_compare_p:nNn { \l__enumext_level_int } = { 1 } }
2072                 }
2073                 {
2074                   \int_gdecr:N \g__enumext_item_number_int
2075                   \bool_set_false:N \l__enumext_item_number_bool
2076                 }
2077             }
2078       }
2079   }
```

(*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.

```
2080 \cs_new_protected:Nn \__enumext_check_ans_key_hook:
2081   {
2082     \bool_lazy_and:nnT
2083       { \bool_if_p:N \l__enumext_check_ans_key_bool }
2084       { \bool_if_p:N \g__enumext_standar_bool }
2085       {
2086         \bool_gset_true:N \g__enumext_check_ans_key_bool
2087       }
2088     \bool_lazy_and:nnT
2089       { \bool_if_p:N \l__enumext_check_ans_key_bool }
2090       { \bool_if_p:N \g__enumext_starred_bool }
2091       {
2092         \bool_gset_true:N \g__enumext_check_ans_key_bool
2093       }
2094   }
```

(*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:.

```
2095  \cs_new_protected:Nn \__enumext_item_answer_diff:
2096    {
2097      \int_gset:Nn \g__enumext_item_answer_diff_int
2098        {
2099          \int_sign:n { \g__enumext_item_number_int - \g__enumext_item_anskey_int }
2100        }
2101    }
```

(*End of definition for* __enumext_item_answer_diff:.)

__enumext_check_ans_show: The function __enumext_check_ans_show: will be executed within the function __enumext_execute_-
__enumext_check_ans_msg_less: after_env: when the key check-ans is active, that is, when \g__enumext_check_ans_key_bool is *"true"*
__enumext_check_ans_msg_same_ok: and will return the appropriate message according to the value of \g__enumext_item_answer_diff_int
__enumext_check_ans_msg_greater: set by the function __enumext_item_answer_diff:.

```
2102  \cs_new_protected:Nn \__enumext_check_ans_show:
2103    {
2104      \int_case:nn { \g__enumext_item_answer_diff_int }
2105        {
2106          { -1 }{ \__enumext_check_ans_msg_less:      }
2107          {  0 }{ \__enumext_check_ans_msg_same_ok: }
2108          {  1 }{ \__enumext_check_ans_msg_greater: }
2109        }
2110    }
2111  \cs_new_protected:Nn \__enumext_check_ans_msg_less:
2112    {
2113      \msg_warning:nneee { enumext } { item-less-answer } { \g__enumext_store_name_tl }
2114        { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2115    }
2116  \cs_new_protected:Nn \__enumext_check_ans_msg_same_ok:
2117    {
2118      \msg_term:nneee { enumext } { items-same-answer } { \g__enumext_store_name_tl }
2119        { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2120    }
2121  \cs_new_protected:Nn \__enumext_check_ans_msg_greater:
2122    {
2123      \msg_warning:nneee { enumext } { item-greater-answer } { \g__enumext_store_name_tl }
2124        { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2125    }
```

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

__enumext_check_ans_log: The function __enumext_check_ans_log: will be executed within the function __enumext_execute_-
__enumext_check_ans_log_msg_less: after_env: when the key check-ans is not active, that is, when \g__enumext_check_ans_key_bool is
__enumext_check_ans_log_msg_same_ok: *"false"* and write in the log the appropriate message according to the value of \g__enumext_item_answer_-
__enumext_check_ans_log_msg_greater: diff_int set by the function __enumext_item_answer_diff:.

```
2126  \cs_new_protected:Nn \__enumext_check_ans_log:
2127    {
2128      \int_case:nn { \g__enumext_item_answer_diff_int }
2129        {
2130          { -1 }{ \__enumext_check_ans_log_msg_less:      }
2131          {  0 }{ \__enumext_check_ans_log_msg_same_ok: }
2132          {  1 }{ \__enumext_check_ans_log_msg_greater: }
2133        }
2134    }
2135  \cs_new_protected:Nn \__enumext_check_ans_log_msg_less:
2136    {
2137      \msg_log:nneee { enumext } { item-less-answer } { \g__enumext_store_name_tl }
2138        { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2139    }
2140  \cs_new_protected:Nn \__enumext_check_ans_log_msg_same_ok:
2141    {
2142      \msg_log:nneee { enumext } { items-same-answer }  { \g__enumext_store_name_tl }
2143        { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2144    }
```

```
2145 \cs_new_protected:Nn \__enumext_check_ans_log_msg_greater:
2146   {
2147     \msg_log:nneee { enumext } { item-greater-answer } { \g__enumext_store_name_tl }
2148       { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2149   }
```

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

12.25.6 Check for \item* and \anspic* commands

__enumext_check_starred_cmd:n

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

```
2150 \cs_new_protected:Npn \__enumext_check_starred_cmd:n #1
2151   {
2152     \int_compare:nNnT
2153       { \g__enumext_check_starred_cmd_int } = { 0 }
2154       {
2155         \msg_warning:nnnV
2156           { enumext } { missing-starred }{ #1 } \l__enumext_check_start_line_env_tl
2157       }
2158     \int_compare:nNnT
2159       { \g__enumext_check_starred_cmd_int } > { 1 }
2160       {
2161         \msg_warning:nnnV
2162           { enumext } { many-starred }{ #1 } \l__enumext_check_start_line_env_tl
2163       }
2164     \int_gzero:N \g__enumext_check_starred_cmd_int
2165     \tl_clear:N \l__enumext_check_start_line_env_tl
2166   }
```

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

12.26 Keys and functions associated with storage

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

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

```
2167 \cs_set_protected:Npn \__enumext_tmp:n #1
2168   {
2169     \keys_define:nn { enumext / #1 }
2170       {
2171         wrap-ans   .cs_set_protected:Np = \__enumext_anskey_wrapper:n ##1,
2172         wrap-ans   .initial:n =
2173                      {
2174                        \fbox{\parbox[t]{\dimeval{\itemwidth -2\fboxsep -2\fboxrule}}{##1}}
2175                      },
2176         wrap-ans   .value_required:n = true,
2177         wrap-opt   .cs_set_protected:Np = \__enumext_keyans_wrapper_opt:n ##1,
2178         wrap-opt   .initial:n = [{##1}],
2179         wrap-opt   .value_required:n = true,
2180         save-sep   .tl_set:N  = \l__enumext_store_keyans_item_opt_sep_tl,
2181         save-sep   .initial:n = {, ~ },
2182         save-sep   .value_required:n = true,
2183         mark-ans   .tl_set:N  = \l__enumext_mark_answer_sym_tl,
2184         mark-ans   .initial:n = \textasteriskcentered,
2185         mark-ans   .value_required:n = true,
2186         mark-pos   .choice:,
2187         mark-pos / left    .code:n = \str_set:Nn \l__enumext_mark_position_str { l },
2188         mark-pos / right   .code:n = \str_set:Nn \l__enumext_mark_position_str { r },
2189         mark-pos / unknown .code:n =
2190                            \msg_error:nneee { enumext } { unknown-choice }
2191                              { mark-pos } { left, ~ right } { \exp_not:n {##1} },
2192         mark-pos   .initial:n = right,
2193         mark-pos   .value_required:n = true,
2194         show-ans   .bool_set:N = \l__enumext_show_answer_bool,
2195         show-ans   .initial:n  = false,
2196         show-ans   .value_required:n = true,
2197         show-pos   .bool_set:N = \l__enumext_show_position_bool,
2198         show-pos   .initial:n  = false,
2199         show-pos   .value_required:n = true,
2200         mark-ref   .tl_set:N   = \l__enumext_mark_ref_sym_tl,
```

```
2201          mark-ref    .initial:n  = \textasteriskcentered,
2202          mark-ref    .value_required:n = true,
2203          save-ref    .bool_set:N = \l__enumext_store_ref_key_bool,
2204          save-ref    .initial:n  = false,
2205          save-ref    .value_required:n = true,
2206       }
2207    }
2208 \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.

```
2209 \cs_set_protected:Npn \__enumext_tmp:n #1
2210    {
2211       \keys_define:nn { enumext / #1 }
2212          {
2213          mark-pos .choice:,
2214          mark-pos / left  .code:n = \str_set:Nn \l__enumext_mark_position_str { l },
2215          mark-pos / right .code:n = \str_set:Nn \l__enumext_mark_position_str { r },
2216          mark-pos .initial:n = right,
2217          mark-pos .value_required:n  = true,
2218          show-ans .bool_set:N = \l__enumext_show_answer_bool,
2219          show-ans .initial:n  = false,
2220          show-ans .value_required:n = true,
2221          show-pos .bool_set:N = \l__enumext_show_position_bool,
2222          show-pos .initial:n  = false,
2223          show-pos .value_required:n = true,
2224          }
2225    }
2226 \clist_map_inline:nn { keyans, keyans* } { \__enumext_tmp:n {#1} }
```

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

12.26.1 Store optional arguments of the environments

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

__enumext_store_active_keys:n
__enumext_store_active_keys_vii:n

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

```
2227 \cs_new_protected:Npn \__enumext_store_active_keys:n #1
2228    {
2229       \bool_if:cF { l__enumext_store_save_key_ \__enumext_level: _bool }
2230          {
2231          \tl_clear:c { l__enumext_save_key_ \__enumext_level: _tl }
2232          \tl_set:ce
2233             { l__enumext_store_save_key_ \__enumext_level: _tl }
2234             { \__enumext_filter_save_key:n {#1} }
2235          }
2236    }
2237 \cs_new_protected:Npn \__enumext_store_active_keys_vii:n #1
2238    {
2239       \bool_if:NF \l__enumext_store_save_key_vii_bool
2240          {
2241          \tl_clear:N \l__enumext_store_save_key_vii_tl
2242          \tl_set:Ne \l__enumext_store_save_key_vii_tl { \__enumext_filter_save_key:n {#1} }
2243          }
2244    }
```

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

12.26.2   Setting save-key key

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

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

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

```
2245 \cs_set_protected:Npn \__enumext_tmp:n #1
2246   {
2247     \keys_define:nn { enumext / enumext* }
2248       {
2249         save-key .code:n = \__enumext_parse_save_key_vii:n {##1},
2250         save-key .value_required:n = true,
2251       }
2252     \keys_define:nn { enumext / #1 }
2253       {
2254         save-key .code:n = \__enumext_parse_save_key:n {##1},
2255         save-key .value_required:n = true,
2256       }
2257   }
2258 \clist_map_inline:nn { level-1, level-2, level-3, level-4 } { \__enumext_tmp:n {#1} }
```

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

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

```
2259 \cs_new_protected:Npn \__enumext_parse_save_key:n #1
2260   {
2261     \bool_set_true:c { l__enumext_store_save_key_ \__enumext_level: _bool }
2262     \tl_clear:c { l__enumext_save_key_ \__enumext_level: _tl }
2263     \tl_set:ce
2264       { l__enumext_store_save_key_ \__enumext_level: _tl }
2265       { \__enumext_filter_save_key:n {#1} }
2266   }
2267 \cs_new_protected:Npn \__enumext_parse_save_key_vii:n #1
2268   {
2269     \bool_set_true:N \l__enumext_store_save_key_vii_bool
2270     \tl_clear:N \l__enumext_store_save_key_vii_tl
2271     \tl_set:Ne \l__enumext_store_save_key_vii_tl { \__enumext_filter_save_key:n {#1} }
2272   }
```

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

12.26.3   Internal functions to store optional arguments

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

```
2273 \cs_new:Npn \__enumext_filter_save_key:n #1
2274   {
2275     \use:e
2276       {
2277         \keyval_parse:NNn
2278           \__enumext_filter_save_key_key:n
2279           \__enumext_filter_save_key_pair:nn {#1}
2280       }
2281   }
```

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

```
2282 \cs_new:Npn \__enumext_filter_save_key_key:n #1
2283   {
2284     \str_case:nnF {#1}
2285       {
2286         { resume } {} { resume* } {} { no-store } {} { base-fix } {}
2287       }
2288       { , { \exp_not:n {#1} } }
2289   }
```

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.

```
2290  \cs_new:Npn \__enumext_filter_save_key_pair:nn #1#2
2291    {
2292      \str_case:nnF {#1}
2293        {
2294          { series   } {} { resume    } {} { save-ans } {} { save-ref   } {}
2295          { save-key } {} { check-ans } {} { show-ans } {} { show-pos   } {}
2296          { wrap-ans } {} { mark-ans  } {} { wrap-opt } {} { save-sep   } {}
2297          { mark-ref } {} { mini-env  } {} { mini-sep } {} { mini-right } {}
2298          { mini-right* } {}
2299        }
2300        { , { \exp_not:n {#1} } = { \exp_not:n {#2} } }
2301    }
```

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

12.26.4   Function for storing content in prop list

`\__enumext_store_addto_prop:n`
`\__enumext_store_addto_prop:V`

The function `\__enumext_store_addto_prop:n` stores the content in ⟨*prop list*⟩ defined by save-ans key. The *"stored content"* is retrieved by means of the `\getkeyans` command.

The form in which the content is *"stored"* in the ⟨*prop list*⟩ is {⟨*position*⟩}{⟨*content*⟩}. This function is used by `\anskey` in enumext and enumext* environments, `\item*` in keyans and keyans* environments and `\anspic*` in keyanspic environment.

```
2302  \cs_new_protected:Npn \__enumext_store_addto_prop:n #1
2303    {
2304      \prop_gput_if_not_in:cen { g__enumext_ \l__enumext_store_name_tl _prop }
2305        {
2306          \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } + 1 }
2307        }
2308        { #1 }
2309    }
2310  \cs_generate_variant:Nn \__enumext_store_addto_prop:n { V, e }
```

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

12.26.5   Function for storing content in sequence

`\__enumext_store_addto_seq:n`
`\__enumext_store_addto_seq:v`
`\__enumext_store_addto_seq:V`

The function `\__enumext_store_addto_seq:n` stores the content in ⟨*sequence*⟩ defined by save-ans key. This function is used by `\anskey` in enumext, `\item*` in keyans and `\anspic` in keyanspic.

The form in which the content is stored in ⟨*sequence*⟩ is in a internal enumext or enumext* environments with the *same structure* in which the command was executed.

The *"stored content"* is retrieved by means of the `\printkeyans` command.

```
2311  \cs_new_protected:Npn \__enumext_store_addto_seq:n #1
2312    {
2313      \seq_gput_right:cn { g__enumext_ \l__enumext_store_name_tl _seq } { #1 }
2314    }
2315  \cs_generate_variant:Nn \__enumext_store_addto_seq:n { v, V, e }
```

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

12.26.6   Functions for storing the list structure in the sequence

`\__enumext_store_level_open:`
`\__enumext_store_level_close:`

The memorization structure of the list is handled by the functions `\__enumext_store_level_open:` and `\__enumext_store_level_close:` which are executed per level within the enumext environment.

```
2316  \cs_new_protected:Nn \__enumext_store_level_open:
2317    {
2318      \bool_if:NT \l__enumext_check_answers_bool
2319        {
2320          \tl_if_empty:cTF { l__enumext_store_save_key_ \__enumext_level: _tl }
2321            {
2322              \__enumext_store_addto_seq:n
2323                {
2324                  \item \begin{enumext}
2325                }
2326            }
2327            {
2328              \tl_put_left:cn { l__enumext_store_save_key_ \__enumext_level: _tl }
2329                {
```

```
2330              \item \begin{enumext} [
2331            }
2332          \tl_put_right:cn { l__enumext_store_save_key_ \__enumext_level: _tl }
2333            {
2334              ]
2335            }
2336          \__enumext_store_addto_seq:v { l__enumext_store_save_key_ \__enumext_level: _tl }
2337        }
2338      }
2339    }
2340  \cs_new_protected:Nn \__enumext_store_level_close:
2341    {
2342      \bool_if:NT \l__enumext_check_answers_bool
2343        {
2344          \__enumext_store_addto_seq:n { \end{enumext} }
2345        }
2346    }
```

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

__enumext_store_level_open_vii:
__enumext_store_level_close_vii:

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

```
2347  \cs_new_protected:Nn \__enumext_store_level_open_vii:
2348    {
2349      \bool_if:NT \l__enumext_check_answers_bool
2350        {
2351          \tl_if_empty:NTF \l__enumext_store_save_key_vii_tl
2352            {
2353              \__enumext_store_addto_seq:n
2354                {
2355                  \item \begin{enumext*}
2356                }
2357            }
2358            {
2359              \tl_put_left:Nn \l__enumext_store_save_key_vii_tl
2360                {
2361                  \item \begin{enumext*}[
2362                }
2363              \tl_put_right:Nn \l__enumext_store_save_key_vii_tl
2364                {
2365                  ]
2366                }
2367              \__enumext_store_addto_seq:V \l__enumext_store_save_key_vii_tl
2368            }
2369        }
2370    }
2371  \cs_new_protected:Nn \__enumext_store_level_close_vii:
2372    {
2373      \bool_if:NT \l__enumext_check_answers_bool
2374        {
2375          \__enumext_store_addto_seq:n { \end{enumext*} }
2376        }
2377    }
```

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

12.26.7 Function for show marks and position

__enumext_print_keyans_box:NN
__enumext_print_keyans_box:cc

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

#1: \l__enumext_labelwidth_X_dim
#2: \l__enumext_labelsep_X_dim

```
2378  \cs_new_protected:Nn \__enumext_print_keyans_box:NN
2379    {
2380      \mode_leave_vertical:
2381      \skip_horizontal:n { -\dim_use:N #2 }
2382      \makebox[0pt][ r ]
2383        {
2384          \makebox[ \dim_use:N #1 ][ \l__enumext_mark_position_str ]
2385            {
2386              \tl_use:N \l__enumext_mark_answer_sym_tl
```

```
2387                    }
2388                }
2389          \skip_horizontal:n { \dim_use:N #2 }
2390      }
2391  \cs_generate_variant:Nn \__enumext_print_keyans_box:NN { cc }
```

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

12.27   The internal label and 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.

__enumext_store_internal_ref:

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.

```
2392  \cs_new_protected:Nn \__enumext_store_internal_ref:
2393    {
2394      \cs_set_protected:Npn \__enumext_tmp:n ##1
2395        {
2396          \tl_set_eq:cc { l__enumext_label_copy_##1_tl } { l__enumext_label_##1_tl }
2397          \tl_reverse:c { l__enumext_label_copy_##1_tl }
2398          \tl_remove_once:cn { l__enumext_label_copy_##1_tl } { . }
2399          \tl_reverse:c { l__enumext_label_copy_##1_tl }
2400        }
2401      \clist_map_inline:nn { i, ii, iii, iv, vii } { \__enumext_tmp:n {##1} }
2402      \cs_set:Npn \__enumext_tmp:n ##1
2403        { . \tl_use:c { l__enumext_label_copy_ \int_to_roman:n {##1} _tl } }
```

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

```
2404          \bool_lazy_all:nT
2405            {
2406              { \bool_if_p:N \g__enumext_starred_bool }
2407              { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
2408            }
2409            {
2410              \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2411                { \tl_use:N \l__enumext_label_copy_vii_tl }
2412            }
2413          \bool_lazy_all:nT
2414            {
2415              { \bool_not_p:n { \g__enumext_standar_bool } }
2416              { \bool_if_p:N \l__enumext_standar_bool }
2417              { \int_compare_p:nNn { \l__enumext_level_int } > { 0 } }
2418            }
2419            {
2420              \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2421                {
2422                  \tl_use:N \l__enumext_label_copy_vii_tl
2423                  \int_step_function:nnN { 1 } { \l__enumext_level_int } \__enumext_tmp:n
2424                }
2425            }
```

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

```
2426          \bool_lazy_all:nT
2427            {
2428              { \bool_if_p:N \g__enumext_standar_bool }
2429              { \int_compare_p:nNn { \l__enumext_level_int } > { 0 } }
2430              { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
2431            }
2432            {
2433              \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2434                {
2435                  \tl_use:N \l__enumext_label_copy_i_tl
2436                  \int_step_function:nnN { 2 } { \l__enumext_level_int } \__enumext_tmp:n
2437                }
2438            }
2439          \cs_set:Npn \__enumext_tmp:n ##1
2440            { \tl_use:c { l__enumext_label_copy_ \int_to_roman:n {##1} _tl } . }
2441          \bool_lazy_all:nT
```

```
2442       {
2443         { \bool_if_p:N \g__enumext_standar_bool }
2444         { \bool_if_p:N \l__enumext_starred_bool }
2445         { \int_compare_p:nNn { \l__enumext_level_int } > { 0 } }
2446       }
2447       {
2448         \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2449           {
2450             \int_step_function:nnN { 1 } { \l__enumext_level_int } \__enumext_tmp:n
2451             \tl_use:N \l__enumext_label_copy_vii_tl
2452           }
2453       }
```

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

```
2454       \tl_put_right:Ne \l__enumext_newlabel_arg_one_tl
2455         {
2456           \l__enumext_store_name_tl \c_colon_str
2457           \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } }
2458         }
```

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.

```
2459       \tl_put_right:Ne \l__enumext_write_aux_file_tl
2460         {
2461           \__enumext_newlabel:nn
2462             { \exp_not:V \l__enumext_newlabel_arg_one_tl }
2463             { \l__enumext_newlabel_arg_two_tl }
2464         }
2465     \l__enumext_write_aux_file_tl
2466   }
```

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

12.28   Common functions for \anskey **and anskey* environment**

__enumext_store_anskey_code:n

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

```
2467 \cs_new_protected:Npn \__enumext_store_anskey_code:n #1
2468   {
2469     \int_gincr:N \g__enumext_item_anskey_int
2470     \__enumext_store_addto_prop:n {#1}
2471     \bool_if:NT \l__enumext_store_ref_key_bool
2472       {
2473         \__enumext_store_internal_ref:
2474       }
2475     \__enumext_anskey_show_wrap_left:n { #1 }
```

Now we start processing the [⟨*key = val*⟩] passed to the command to build our \item in the variable \l__enumext_store_anskey_arg_tl which we will *"store"* in the ⟨*sequence*⟩. First we clear the variable \l__enumext_store_anskey_arg_tl and process the ⟨*keys*⟩, if the break-col key is present and the command is running under enumext (not in enumext*) we will add \columnbreak and then \item.

```
2476     \tl_clear:N \l__enumext_store_anskey_arg_tl
2477     \bool_lazy_and:nnT
2478       { \bool_if_p:N \l__enumext_store_columns_break_bool }
2479       { \bool_not_p:n { \l__enumext_starred_bool } }
2480       {
2481         \tl_put_left:Nn \l__enumext_store_anskey_arg_tl { \columnbreak }
2482       }
2483     \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.

```
2484       \bool_lazy_and:nnT
2485         { \bool_not_p:n { \l__enumext_starred_bool } }
2486         { \int_compare_p:nNn { \l__enumext_store_item_join_int } > { 1 } }
2487         {
2488           \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2489             {
2490               ( \exp_not:V \l__enumext_store_item_join_int )
2491             }
2492         }
```

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

```
2493        \bool_if:NTF \l__enumext_store_item_star_bool
2494          {
2495            \tl_put_right:Nn \l__enumext_store_anskey_arg_tl { * }
2496            \tl_if_empty:NF \l__enumext_store_item_symbol_tl
2497              {
2498                \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2499                  {
2500                    [ \exp_not:V \l__enumext_store_item_symbol_tl ]
2501                  }
2502              }
2503            \dim_compare:nT
2504              {
2505                \l__enumext_store_item_symbol_sep_dim != \c_zero_dim
2506              }
2507              {
2508                \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2509                  {
2510                    [ \exp_not:V \l__enumext_store_item_symbol_sep_dim ]
2511                  }
2512              }
2513            \tl_put_right:Nn \l__enumext_store_anskey_arg_tl {#1}
2514          }
2515          {
2516            \tl_put_right:Nn \l__enumext_store_anskey_arg_tl {#1}
2517          }
```

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*⟩.

```
2518        \bool_lazy_and:nnT
2519          { \bool_if_p:N \l__enumext_store_ref_key_bool }
2520          { \bool_if_p:N \l__enumext_hyperref_bool }
2521          {
2522            \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2523              {
2524                \hfill \exp_not:N \hyperlink { \exp_not:V \l__enumext_newlabel_arg_one_tl }
2525                  { \exp_not:V \l__enumext_mark_ref_sym_tl }
2526              }
2527          }
2528        \__enumext_store_addto_seq:V \l__enumext_store_anskey_arg_tl
2529      }
```

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

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

```
2530    \cs_new_protected:Npn \__enumext_anskey_show_wrap_arg:n #1
2531      {
2532        \par
2533        \bool_if:NTF \l__enumext_starred_bool
2534          {
2535            \__enumext_print_keyans_box:NN \l__enumext_labelwidth_vii_dim  \l__enumext_labelsep_vii_d
2536          }
2537          {
2538            \__enumext_print_keyans_box:cc
2539              { l__enumext_labelwidth_ \__enumext_level: _dim }
2540              { l__enumext_labelsep_ \__enumext_level: _dim }
2541          }
2542        \__enumext_anskey_wrapper:n { #1 }
2543      }
```

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

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

```
2544    \cs_new_protected:Npn \__enumext_anskey_show_wrap_left:n #1
2545      {
2546        \bool_if:NT \l__enumext_show_answer_bool
```

```
2547        {
2548            \__enumext_anskey_show_wrap_arg:n { #1 }
2549        }
2550      \bool_if:NT \l__enumext_show_position_bool
2551        {
2552          \tl_set:Ne \l__enumext_mark_answer_sym_tl
2553            {
2554              \group_begin:
2555              \exp_not:N \normalfont
2556              \exp_not:N \footnotesize [ \int_eval:n
2557                {
2558                  \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
2559                }
2560                ]
2561              \group_end:
2562            }
2563            \__enumext_anskey_show_wrap_arg:n { #1 }
2564        }
2565    }
```

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

12.29   The command \anskey

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*⟩}.

__enumext_anskey_unknown:n
__enumext_anskey_unknown:nn

First we'll add the keys break-col, item-join, item-star, item-sym* and item-pos*.

```
2566  \keys_define:nn { enumext / anskey }
2567    {
2568      break-col .bool_set:N = \l__enumext_store_columns_break_bool,
2569      break-col .default:n  = true,
2570      break-col .value_forbidden:n = true,
2571      item-join .int_set:N  = \l__enumext_store_item_join_int,
2572      item-join .value_required:n = true,
2573      item-star .bool_set:N = \l__enumext_store_item_star_bool,
2574      item-star .default:n  = true,
2575      item-star .value_forbidden:n = true,
2576      item-sym* .tl_set:N   = \l__enumext_store_item_symbol_tl,
2577      item-sym* .value_required:n = true,
2578      item-pos* .dim_set:N  = \l__enumext_store_item_symbol_sep_dim,
2579      item-pos* .value_required:n = true,
2580      unknown   .code:n     = { \__enumext_anskey_unknown:n {#1} },
2581    }
```

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

```
2582  \cs_new_protected:Npn \__enumext_anskey_unknown:n #1
2583    {
2584      \exp_args:NV \__enumext_anskey_unknown:nn \l_keys_key_str {#1}
2585    }
2586  \cs_new_protected:Npn \__enumext_anskey_unknown:nn #1 #2
2587    {
2588      \tl_if_blank:nTF {#2}
2589        {
2590          \msg_error:nnn { enumext } { anskey-cmd-key-unknown } {#1}
2591        }
2592        {
2593          \msg_error:nnnn { enumext } { anskey-cmd-key-value-unknown } {#1} {#2}
2594        }
2595    }
```

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

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

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

that the argument is not empty, finally search the [⟨*key = val*⟩] and call the function __enumext_store_-
anskey_code:n.

```
2596 \NewDocumentCommand \anskey { o +m }
2597   {
2598     \__enumext_anskey_safe_outer:
2599     \group_begin:
2600       \bool_if:NT \l__enumext_check_answers_bool
2601         {
2602           \tl_if_novalue:nF {#1}
2603             {
2604               \keys_set:nn { enumext / anskey } {#1}
2605             }
2606           \tl_if_blank:nTF {#2}
2607             {
2608               \msg_error:nn { enumext } { anskey-empty-arg }
2609             }
2610             {
2611               \__enumext_anskey_safe_inner:
2612               \__enumext_store_anskey_code:n {#2}
2613             }
2614         }
2615     \group_end:
2616   }
```

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

12.29.1   Internal functions for the command

__enumext_anskey_safe_outer:
__enumext_anskey_safe_inner:

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

```
2617 \cs_new_protected:Nn \__enumext_anskey_safe_outer:
2618   {
2619     \bool_if:NF \l__enumext_store_active_bool
2620       {
2621         \msg_error:nnnn { enumext } { anskey-wrong-place }{ anskey }{ enumext }
2622       }
2623     \int_compare:nNnT { \l__enumext_keyans_level_int } = { 1 }
2624       {
2625         \msg_error:nnnn { enumext } { command-wrong-place }{ anskey }{ keyans }
2626       }
2627     \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
2628       {
2629         \msg_error:nnnn { enumext } { command-wrong-place }{ anskey }{ keyans* }
2630       }
2631     \int_compare:nNnT { \l__enumext_keyans_pic_level_int } = { 1 }
2632       {
2633         \msg_error:nnnn { enumext } { command-wrong-place }{ anskey }{ keyanspic }
2634       }
2635   }
```

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

```
2636 \cs_new_protected:Nn \__enumext_anskey_safe_inner:
2637   {
2638     \int_incr:N \l__enumext_anskey_level_int
2639     \int_compare:nNnT { \l__enumext_anskey_level_int } > { 1 }
2640       {
2641         \msg_error:nn { enumext } { anskey-nested }
2642       }
2643     \bool_if:NF \l__enumext_item_number_bool
2644       {
2645         \msg_error:nn { enumext } { anskey-unnumber-item }
2646       }
2647     \mode_if_math:T
2648       {
2649         \msg_error:nne { enumext } { anskey-math-mode } { \c_backslash_str anskey }
2650       }
2651   }
```

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

12.30 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: (§12.31) which is executed after the environment in which the key save-ans is active.

```
2652 \cs_new_protected:Nn \__enumext_undefine_anskey_env:
2653   {
2654     \cs_undefine:c { anskey* }
2655     \cs_undefine:c { endanskey* }
2656     \cs_undefine:c { __scontents_anskey*_env_begin: }
2657     \cs_undefine:c { __scontents_anskey*_env_end: }
2658   }
```

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

```
2659 \__enumext_before_env:nn { enumext }
2660   {
2661     \bool_lazy_and:nnT
2662       { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
2663       { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
2664       {
2665         \cs_if_free:cF { __scontents_anskey*_env_begin: }
2666           {
2667             \msg_error:nnn { enumext } { anskey-env-error } { anskey* }
2668           }
2669       }
2670   }
2671 \__enumext_before_env:nn { enumext* }
2672   {
2673     \bool_lazy_and:nnT
2674       { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
2675       { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
2676       {
2677         \cs_if_free:cF { __scontents_anskey*_env_begin: }
2678           {
2679             \msg_error:nnn { enumext } { anskey-env-error } { anskey* }
2680           }
2681       }
2682   }
```

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

```
2683 \__enumext_before_env:nn { anskey* }
2684   {
2685     \int_compare:nNnT { \l__enumext_keyans_level_int } = { 1 }
2686       {
2687         \msg_error:nnn { enumext } { anskey-env-wrong }{ keyans }
2688       }
2689     \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
2690       {
2691         \msg_error:nnn { enumext } { anskey-env-wrong } { keyans* }
2692       }
2693     \int_compare:nNnT { \l__enumext_keyans_pic_level_int } = { 1 }
2694       {
2695         \msg_error:nnn { enumext } { anskey-env-wrong } { keyanspic }
2696       }
2697     \bool_if:NF \l__enumext_item_number_bool
2698       {
```

```
2699                    \msg_error:nn { enumext } { anskey-unnumber-item }
2700                  }
2701              \mode_if_math:T
2702                  {
2703                    \msg_error:nnn { enumext } { anskey-math-mode } { anskey* }
2704                  }
2705          }
```

(*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: (§12.25.1) and we will execute it only if the variable \l__enumext_anskey_env_bool is true, with this we prevent it from being executed again when the environment is nested and the key save-ans is active, which returns an error for part of the package scontents.

```
2706  \cs_new_protected:Npn \__enumext_anskey_env_make:n #1
2707      {
2708        \bool_if:NT \l__enumext_anskey_env_bool
2709            {
2710              \newenvsc{anskey*}[store-env=#1,print-env=false]
2711              \__enumext_anskey_env_exec:
2712            }
2713      }
2714  \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.

```
2715  \cs_new_protected:Nn \__enumext_anskey_env_define_keys:
2716      {
2717        \keys_define:nn { scontents / scontents }
2718            {
2719              break-col .bool_gset:N = \g__enumext_store_columns_break_bool,
2720              break-col .default:n   = true,
2721              break-col .value_forbidden:n = true,
2722              item-join .int_gset:N  = \g__enumext_store_item_join_int,
2723              item-join .value_required:n = true,
2724              item-star .bool_gset:N = \g__enumext_store_item_star_bool,
2725              item-star .default:n   = true,
2726              item-star .value_forbidden:n = true,
2727              item-sym* .tl_gset:N   = \g__enumext_store_item_symbol_tl,
2728              item-sym* .value_required:n = true,
2729              item-pos* .dim_gset:N  = \g__enumext_store_item_symbol_sep_dim,
2730              item-pos* .value_required:n = true,
2731              print-env .undefine:,
2732              store-env .undefine:,
2733              write-out .undefine:,
2734              unknown   .code:n      = { \__enumext_anskey_env_unknown:n {##1} },
2735            }
2736      }
```

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

```
2737  \cs_new_protected:Npn \__enumext_anskey_env_unknown:n #1
2738      {
2739        \exp_args:NV \__enumext_anskey_env_unknown:nn \l_keys_key_str {#1}
2740      }
2741  \cs_new_protected:Npn \__enumext_anskey_env_unknown:nn #1#2
2742      {
2743        \tl_if_blank:nTF {#2}
2744            {
2745              \msg_error:nnn { enumext } { anskey-env-key-unknown } {#1}
2746            }
2747            {
2748              \msg_error:nnnn { enumext } { anskey-env-key-value-unknown } {#1} {#2}
2749            }
2750      }
```

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

```
2751 \cs_new_protected:Nn \__enumext_anskey_env_reset_keys:
2752   {
2753     \keys_define:nn { scontents / scontents }
2754       {
2755         break-col .undefine:,
2756         item-join .undefine:,
2757         item-star .undefine:,
2758         item-sym* .undefine:,
2759         item-pos* .undefine:,
2760         write-out .code:n    = {
2761                               \bool_set_false:N \l__scontents_storing_bool
2762                               \bool_set_true:N  \l__scontents_writing_bool
2763                               \tl_set:Nn \l__scontents_fname_out_tl {##1}
2764                               },
2765         write-out .value_required:n = true,
2766         print-env .meta:nn   = { scontents } { print-env = ##1 },
2767         print-env .default:n = true,
2768         store-env .meta:nn   = { scontents } { store-env = ##1 },
2769         unknown   .code:n    = { \__scontents_parse_environment_keys:n {##1} },
2770       }
2771   }
```

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*.

```
2772 \cs_new_protected:Npn \__enumext_rescan_anskey_env:n #1
2773   {
2774     \group_begin:
2775       \int_set:Nn \tex_newlinechar:D { `\^^J }
2776       \__scontents_rescan_tokens:x
2777         {
2778           \endgroup % This assumes \catcode`\\=0... Things might go off otherwise.
2779           #1
2780         }
2781   }
```

(*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*⟩.

```
2782 \cs_new_protected:Nn \__enumext_anskey_env_exec:
2783   {
2784     \__enumext_before_env:nn { anskey* }
2785       {
2786         \__enumext_anskey_env_define_keys:
2787       }
```

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.

```
2788     \hook_if_empty:nF {env/anskey*/after}
2789       {
2790         \hook_gremove_code:nn {env/anskey*/after} { * }
2791       }
2792     \__enumext_after_env:nn { anskey* }
2793       {
2794         \__enumext_anskey_env_save_keys:
2795         \tl_clear:N \l__enumext_store_anskey_env_tl
2796         \tl_clear:N \l__enumext_store_anskey_opt_tl
2797         \bool_if:NT \l__enumext_check_answers_bool
2798           {
2799             \tl_gset:Ne \l__enumext_store_anskey_env_tl
2800               {
2801                 \seq_item:ce { g__scontents_name_ \l__enumext_store_name_tl _seq } { -1 }
2802               }
2803             \regex_match:nVTF
2804               { ^\s* \z | ^\s* \u{c__scontents_hidden_space_str} \z }
2805               \l__enumext_store_anskey_env_tl
```

```
2806                    {
2807                        \msg_error:nn { enumext } { anskey-empty-arg }
2808                    }
2809                    {
2810                        \__enumext_anskey_env_store:
2811                    }
2812                }
2813            \__enumext_anskey_env_clean_vars:
2814            \__enumext_anskey_env_reset_keys:
2815        }
2816    }
```

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

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

__enumext_anskey_env_save_keys:
__enumext_anskey_env_store:
__enumext_anskey_env_clean_vars:

The function __enumext_anskey_env_save_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.

```
2817  \cs_new_protected:Nn \__enumext_anskey_env_save_keys:
2818    {
2819        \bool_lazy_and:nnT
2820            { \bool_if_p:N \g__enumext_store_columns_break_bool }
2821            { \bool_not_p:n { \l__enumext_starred_bool } }
2822            {
2823                \tl_put_left:Ne \l__enumext_store_anskey_opt_tl { ,break-col, }
2824            }
```

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

```
2825        \bool_lazy_and:nnT
2826            { \bool_not_p:n { \l__enumext_starred_bool } }
2827            { \int_compare_p:nNn { \g__enumext_store_item_join_int } > { 1 } }
2828            {
2829                \tl_put_left::Ne \l__enumext_store_anskey_opt_tl
2830                    {
2831                        ,item-join = \exp_not:V \g__enumext_store_item_join_int,
2832                    }
2833            }
```

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

```
2834        \bool_if:NT \g__enumext_store_item_star_bool
2835            {
2836                \tl_put_left:Ne \l__enumext_store_anskey_opt_tl
2837                    {
2838                        ,item-star,
2839                    }
2840                \tl_if_empty:NF \g__enumext_store_item_symbol_tl
2841                    {
2842                        \tl_put_left:Ne \l__enumext_store_anskey_opt_tl
2843                            {
2844                                ,item-sym* = \exp_not:V \g__enumext_store_item_symbol_tl,
2845                            }
2846                    }
2847                \dim_compare:nT
2848                    {
2849                        \g__enumext_store_item_symbol_sep_dim != \c_zero_dim
2850                    }
2851                    {
2852                        \tl_put_left:Ne \l__enumext_store_anskey_opt_tl
2853                            {
2854                                ,item-pos* = \exp_not:V \g__enumext_store_item_symbol_sep_dim,
2855                            }
2856                    }
2857            }
2858    }
```

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

```
2859  \cs_new_protected:Nn \__enumext_anskey_env_store:
```

```
2860    {
2861      \group_begin:
2862        \tl_if_empty:NTF \l__enumext_store_anskey_opt_tl
2863          {
2864            \exp_args:Ne
2865              \__enumext_store_anskey_code:n
2866                {
2867                  \__enumext_rescan_anskey_env:n { \l__enumext_store_anskey_env_tl }
2868                }
2869          }
2870          {
2871            \keys_set_known:nV { enumext / anskey } \l__enumext_store_anskey_opt_tl
2872            \exp_args:Ne
2873              \__enumext_store_anskey_code:n
2874                {
2875                  \__enumext_rescan_anskey_env:n { \l__enumext_store_anskey_env_tl }
2876                }
2877          }
2878      \group_end:
2879    }
```

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

```
2880  \cs_new_protected:Nn \__enumext_anskey_env_clean_vars:
2881    {
2882      \bool_gset_false:N \g__enumext_store_columns_break_bool
2883      \int_gzero:N       \g__enumext_store_item_join_int
2884      \bool_gset_false:N \g__enumext_store_item_star_bool
2885      \tl_gclear:N       \g__enumext_store_item_symbol_tl
2886      \dim_gzero:N       \g__enumext_store_item_symbol_sep_dim
2887    }
```

(*End of definition for* __enumext_anskey_env_save_keys: , __enumext_anskey_env_store: , *and* __enumext_anskey_env_clean_vars: .)

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

__enumext_execute_after_env:

The __enumext_execute_after_env: function will first return the appropriate message for the end of the environment in which the save-ans key is being executed, then call the __enumext_item_answer_diff: function and then will write the values of the global variables used to the .log file. If the key check-ans is active it will execute the function __enumext_check_ans_show: and show the result in the terminal, otherwise it will execute the function __enumext_check_ans_log: and write the results in the .log file, undefine the environment anskey* (§12.30) 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.

```
2888  \cs_new_protected:Nn \__enumext_execute_after_env:
2889    {
2890      \int_compare:nNnT { \l__enumext_level_int } = { 0 }
2891        {
2892          \tl_if_empty:NF \g__enumext_store_name_tl
2893            {
2894              \__enumext_stop_save_ans_msg:
2895              \__enumext_item_answer_diff:
2896              \__enumext_log_global_vars:
2897              \__enumext_log_answer_vars:
2898              \bool_if:NTF \g__enumext_check_ans_key_bool
2899                {
2900                  \__enumext_check_ans_show:
2901                }
2902                { \__enumext_check_ans_log: }
2903              \__enumext_undefine_anskey_env:
2904            }
2905          \__enumext_reset_global_vars:
2906        }
2907    }
```

(*End of definition for* __enumext_execute_after_env: .)

🎯 This function is passed to the function __enumext_after_env:nn for the environments enumext (§12.38) and enumext* (§12.43) and it is executed only when the environments are not nested or at some level of these..

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

12.32.1 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`.

```
2908 \cs_new_protected:Npn \__enumext_keyans_addto_prop:n #1
2909   {
2910     \tl_clear:N \l__enumext_store_current_label_tl
2911     \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
2912       {
2913         \tl_put_right:Ne \l__enumext_store_current_label_tl { \l__enumext_label_vi_tl }
2914       }
2915       {
2916         \tl_put_right:Ne \l__enumext_store_current_label_tl { \l__enumext_label_v_tl }
2917       }
2918     \tl_if_novalue:nF { #1 }
2919       {
2920         % Set save-sep
2921         \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_tl
2922           {
2923             \tl_put_right:Ne \l__enumext_store_current_label_tl { \l__enumext_store_keyans_item_o|
2924           }
2925         \tl_put_right:Ne \l__enumext_store_current_label_tl { #1 }
2926       }
2927     \__enumext_store_addto_prop:V \l__enumext_store_current_label_tl
2928   }
```

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

12.32.2 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.

```
2929 \cs_new_protected:Nn \__enumext_keyans_store_ref:
2930   {
2931     \bool_if:NT \l__enumext_store_ref_key_bool
2932       {
2933         \cs_set_protected:Npn \__enumext_tmp:n ##1
2934           {
2935             \tl_set_eq:cc { l__enumext_label_copy_##1_tl } { l__enumext_label_##1_tl }
2936             \tl_reverse:c { l__enumext_label_copy_##1_tl }
2937             \tl_remove_once:cn { l__enumext_label_copy_##1_tl } { . }
2938             \tl_reverse:c { l__enumext_label_copy_##1_tl }
2939           }
2940         \clist_map_inline:nn { i, v, vi, vii, viii } { \__enumext_tmp:n {##1} }
2941         \__enumext_keyans_store_ref_aux_i:
2942       }
2943   }
```

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`.

```
2944 \cs_new_protected:Nn \__enumext_keyans_store_ref_aux_i:
2945   {
2946     \bool_if:NT \g__enumext_starred_bool
2947       {
2948         \tl_set_eq:NN \l__enumext_label_copy_i_tl \l__enumext_label_copy_vii_tl
2949       }
2950     \int_compare:nNnT { \l__enumext_keyans_pic_level_int } = { 1 }
2951       {
2952         \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2953           { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_vi_tl }
```

```
2954          }
2955      \int_compare:nNnT { \l__enumext_keyans_level_int } = { 1 }
2956        {
2957          \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2958            { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_v_tl }
2959        }
2960      \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
2961        {
2962          \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2963            { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_viii_tl }
2964        }
2965      \tl_put_right:Ne \l__enumext_newlabel_arg_one_tl
2966        {
2967          \l__enumext_store_name_tl \c_colon_str
2968          \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } }
2969        }
2970      \__enumext_keyans_store_ref_aux_ii:
2971    }
```

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.

```
2972  \cs_new_protected:Nn \__enumext_keyans_store_ref_aux_ii:
2973    {
2974      \tl_put_right:Ne \l__enumext_write_aux_file_tl
2975        {
2976          \__enumext_newlabel:nn
2977            { \exp_not:V \l__enumext_newlabel_arg_one_tl }
2978            { \l__enumext_newlabel_arg_two_tl }
2979        }
2980      \l__enumext_write_aux_file_tl
2981    }
```

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

12.32.3 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.

```
2982  \cs_new_protected:Npn \__enumext_keyans_addto_seq:n #1
2983    {
2984      \tl_clear:N \l__enumext_store_current_label_tl
2985      \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
2986        {
2987          \tl_put_right:Ne \l__enumext_store_current_label_tl { \item \l__enumext_label_vi_tl }
2988        }
2989        {
2990          \tl_put_right:Ne \l__enumext_store_current_label_tl { \item \l__enumext_label_v_tl }
2991        }
2992      \tl_if_novalue:nF { #1 }
2993        {
2994          \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_tl
2995            {
2996              \tl_put_right:Ne \l__enumext_store_current_label_tl
2997                {
2998                  \l__enumext_store_keyans_item_opt_sep_tl
2999                }
3000            }
3001          \tl_put_right:Ne \l__enumext_store_current_label_tl { #1 }
3002        }
3003      \__enumext_keyans_addto_seq_link:
3004    }
```

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.

```
3005  \cs_new_protected:Nn \__enumext_keyans_addto_seq_link:
3006    {
3007      \bool_lazy_and:nnT
3008        { \bool_if_p:N \l__enumext_store_ref_key_bool }
3009        { \bool_if_p:N \l__enumext_hyperref_bool }
3010        {
3011          \tl_put_right:Ne \l__enumext_store_current_label_tl
3012            {
3013              \hfill \exp_not:N \hyperlink
3014                {
3015                  \exp_not:V \l__enumext_newlabel_arg_one_tl
3016                }
3017                { \exp_not:V \l__enumext_mark_ref_sym_tl }
3018            }
3019        }
3020      \__enumext_store_addto_seq:V \l__enumext_store_current_label_tl
3021      \bool_if:NT \l__enumext_check_answers_bool
3022        {
3023          \int_gincr:N \g__enumext_item_anskey_int
3024        }
3025    }
```

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

12.32.4 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.

```
3026  \cs_new_protected:Npn \__enumext_keyans_show_left:n #1
3027    {
3028      \tl_if_novalue:nF { #1 }
3029        {
3030          \tl_set:Ne \l__enumext_store_current_opt_arg_tl { #1 }
3031        }
3032      \bool_if:NT \l__enumext_show_answer_bool
3033        {
3034          \__enumext_keyans_show_ans:
3035        }
3036      \bool_if:NT \l__enumext_show_position_bool
3037        {
3038          \__enumext_keyans_show_pos:
3039        }
3040    }
3041  \cs_new_protected:Nn \__enumext_keyans_show_item_opt:
3042    {
3043      \tl_if_empty:NF \l__enumext_store_current_opt_arg_tl
3044        {
3045          \bool_lazy_or:nnT
3046            { \bool_if_p:N \l__enumext_show_answer_bool }
3047            { \bool_if_p:N \l__enumext_show_position_bool }
3048            {
3049              \__enumext_keyans_wrapper_opt:n { \l__enumext_store_current_opt_arg_tl } \c_space_tl
3050            }
3051        }
3052    }
3053  \cs_new_protected:Nn \__enumext_keyans_show_ans:
3054    {
3055      \bool_if:NT \l__enumext_starred_bool
3056        {
3057          \dim_set_eq:NN \l__enumext_labelwidth_i_dim \l__enumext_labelwidth_vii_dim
3058          \dim_set_eq:NN \l__enumext_labelsep_i_dim \l__enumext_labelsep_vii_dim
3059        }
3060      \tl_put_left:Nn \l__enumext_label_v_tl
3061        {
3062          \__enumext_print_keyans_box:NN
3063            \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
3064        }
3065    }
3066  \cs_new_protected:Nn \__enumext_keyans_show_pos:
```

```
3067     {
3068       \bool_if:NT \l__enumext_starred_bool
3069         {
3070           \dim_set_eq:NN \l__enumext_labelwidth_i_dim \l__enumext_labelwidth_vii_dim
3071           \dim_set_eq:NN \l__enumext_labelsep_i_dim \l__enumext_labelsep_vii_dim
3072         }
3073       \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
3074         {
3075           \tl_set:Ne \l__enumext_mark_answer_sym_tl
3076             {
3077               \group_begin:
3078               \exp_not:N \normalfont
3079               \exp_not:N \footnotesize [ \int_eval:n
3080                 {
3081                   \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
3082                 }
3083                 ]
3084               \group_end:
3085             }
3086         }
3087         {
3088           \tl_set:Ne \l__enumext_mark_answer_sym_tl
3089             {
3090               \group_begin:
3091               \exp_not:N \normalfont
3092               \exp_not:N \footnotesize [ \int_eval:n
3093                 {
3094                   \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } + 1
3095                 }
3096                 ]
3097               \group_end:
3098             }
3099         }
3100       \tl_put_left:Nn \l__enumext_label_v_tl
3101         {
3102           \__enumext_print_keyans_box:NN
3103             \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
3104         }
3105     }
```

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

12.33   Redefining `\item` and `\makelabel` in enumext

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.

The `\item` and `\item[`⟨*custom*⟩`]` commands work in the usual way on enumext and we will add `\item*`, `\item*[`⟨*symbol*⟩`]` and `\item*[`⟨*symbol*⟩`][`⟨*offset*⟩`]`.

`\__enumext_default_item:n` 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_answers_bool` set by the key `no-store`, set the boolean variable `\l__enumext_-wrap_label_X_bool` to "true" for the key `wrap-label` and execute `\__enumext_item_std:w` and the key `itemindent`, 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 and the key `itemindent`.

```
3106  \cs_new_protected:Npn \__enumext_default_item:n #1
3107    {
3108      \tl_if_novalue:nTF {#1}
3109        {
3110          \bool_if:NT \l__enumext_check_answers_bool
3111            {
3112              \int_gincr:N \g__enumext_item_number_int
3113              \bool_set_true:N \l__enumext_item_number_bool
3114            }
3115          \bool_set_true:c { l__enumext_wrap_label_ \__enumext_level: _bool }
3116          \__enumext_item_std:w \tl_use:c { l__enumext_fake_item_indent_ \__enumext_level: _tl }
3117        }
3118        {
3119          \bool_set_eq:cc
3120            { l__enumext_wrap_label_ \__enumext_level: _bool }
```

```
3121                 { l__enumext_wrap_label_opt_ \__enumext_level: _bool }
3122             \__enumext_item_std:w [#1] \tl_use:c { l__enumext_fake_item_indent_ \__enumext_level: _tl
3123         }
3124     }
```

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

`\__enumext_starred_item:nn`
`\__enumext_item_star_exec:`

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 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 *"first"* optional argument in the global variable \g__enumext_item_symbol_aux_tl, followed by setting the variable \l__enumext_item_symbol_sep_X_dim set by the key item-pos* or by the *"second"* optional argument, then we will see the state of the variable \l__enumext_check_answers_bool set by the key no-store, set the boolean variable \l__enumext_wrap_label_X_bool to "true" for the key wrap-label and execute __enumext_item_std:w and the key itemindent.

```
3125 \cs_new_protected:Npn \__enumext_starred_item:nn #1 #2
3126     {
3127         \tl_if_novalue:nTF {#1}
3128             {
3129                 \tl_gset_eq:Nc
3130                     \g__enumext_item_symbol_aux_tl { l__enumext_item_symbol_ \__enumext_level: _tl }
3131             }
3132             {
3133                 \tl_gset:Nn \g__enumext_item_symbol_aux_tl {#1}
3134             }
3135         \tl_if_novalue:nTF {#2}
3136             {
3137                 \dim_set_eq:cc
3138                     { l__enumext_item_symbol_sep_ \__enumext_level: _dim }
3139                     { l__enumext_labelsep_ \__enumext_level: _dim }
3140             }
3141             {
3142                 \dim_set:cn { l__enumext_item_symbol_sep_ \__enumext_level: _dim } {#2}
3143             }
3144         \bool_if:NT \l__enumext_check_answers_bool
3145             {
3146                 \int_gincr:N \g__enumext_item_number_int
3147                 \bool_set_true:N \l__enumext_item_number_bool
3148             }
3149         \bool_set_true:c { l__enumext_wrap_label_ \__enumext_level: _bool }
3150         \__enumext_item_std:w \tl_use:c { l__enumext_fake_item_indent_ \__enumext_level: _tl }
3151     }
```

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

```
3152 \cs_new_protected:Nn \__enumext_item_star_exec:
3153     {
3154         \tl_if_empty:cF { l__enumext_item_symbol_ \__enumext_level: _tl }
3155             {
3156                 \mode_leave_vertical:
3157                 \skip_horizontal:n { -\dim_use:c { l__enumext_item_symbol_sep_ \__enumext_level: _dim } }
3158                 \hbox_overlap_left:n { \g__enumext_item_symbol_aux_tl }
3159                 \skip_horizontal:n { \dim_use:c { l__enumext_item_symbol_sep_ \__enumext_level: _dim } }
3160             }
3161     }
```

(*End of definition for* `\__enumext_starred_item:nn` *and* `\__enumext_item_star_exec:`.)

`\__enumext_redefine_item:`

The function __enumext_redefine_item: will redefine the \item command in the enumext environment adding \item*.

```
3162 \cs_new_protected:Nn \__enumext_redefine_item:
3163     {
3164         \RenewDocumentCommand \item { s o o }
3165             {
3166                 \bool_if:nTF {##1}
3167                     {
3168                         \__enumext_starred_item:nn {##2} {##3}
```

```
3169                    }
3170                { \__enumext_default_item:n {##2} }
3171            }
3172        }
```

(*End of definition for* __enumext_redefine_item:.)

__enumext_make_label:
__enumext_make_label_std:
__enumext_make_label_box:

The function __enumext_make_label: redefine \makelabel for the keys align, font, wrap-label, wrap-label* and \item* for enumext environment.

```
3173 \cs_new_protected:Nn \__enumext_make_label:
3174    {
3175        \IfDocumentMetadataTF
3176            {
3177                \__enumext_make_label_box:
3178            }
3179            { \__enumext_make_label_std: }
3180    }
3181 \cs_new_protected:Nn \__enumext_make_label_std:
3182    {
3183        \RenewDocumentCommand \makelabel { m }
3184            {
3185                \tl_use:c { l__enumext_label_fill_left_ \__enumext_level: _tl }
3186                \tl_use:c { l__enumext_label_font_style_ \__enumext_level: _tl }
3187                \bool_if:cTF { l__enumext_wrap_label_ \__enumext_level: _bool }
3188                    {
3189                        \__enumext_item_star_exec:
3190                        \use:c { __enumext_wrapper_label_ \__enumext_level: :n } { ##1 }
3191                    }
3192                    { ##1 }
3193                \tl_use:c { l__enumext_label_fill_right_ \__enumext_level: _tl }
3194                \tl_gclear:N \g__enumext_item_symbol_aux_tl
3195            }
3196    }
3197 \cs_new_protected:Nn \__enumext_make_label_box:
3198    {
3199        \RenewDocumentCommand \makelabel { m }
3200            {
3201                \makebox
3202                [ \dim_use:c { l__enumext_labelwidth_ \__enumext_level: _dim } ]
3203                [ \str_use:c { l__enumext_align_label_pos_ \__enumext_level: _str } ]
3204                    {
3205                        \tl_use:c { l__enumext_label_font_style_ \__enumext_level: _tl }
3206                        \bool_if:cTF { l__enumext_wrap_label_ \__enumext_level: _bool }
3207                            {
3208                                \__enumext_item_star_exec:
3209                                \use:c { __enumext_wrapper_label_ \__enumext_level: :n } { ##1 }
3210                            }
3211                            { ##1 }
3212                        \tl_gclear:N \g__enumext_item_symbol_aux_tl
3213                    }
3214            }
3215    }
```

(*End of definition for* __enumext_make_label:, __enumext_make_label_std:, *and* __enumext_make_label_box:.)

🎯 This functions are passed to __enumext_list_arg_two_X: used in the definition of the enumext environment (§12.38).

12.34   Setting item-sym* and item-pos* keys

In order to have a cleaner implementation of \item* for the enumext and enumext* environments it is best to define a couple of keys that allow us to control and set by default the ⟨*symbol*⟩ and its ⟨*offset*⟩.

item-sym*
item-pos*

Define and set item-sym* and item-pos* keys for enumext and enumext*.

```
3216 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
3217    {
3218        \keys_define:nn { enumext / #1 }
3219            {
3220                item-sym* .tl_set:c  = { l__enumext_item_symbol_#2_tl },
3221                item-sym* .value_required:n = true,
3222                item-sym* .initial:n  = {$\star$},
3223                item-pos* .dim_set:c  = { l__enumext_item_symbol_sep_#2_dim },
3224                item-pos* .value_required:n = true,
3225            }
```

```
3226      }
3227  \clist_map_inline:nn
3228      {
3229        {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {enumext*}{vii}
3230      }
3231      { \__enumext_tmp:nn #1 }
```

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

12.35   Handling unknown keys

At this point in the code I already know that I will not add more ⟨*keys*⟩ and since I have already been quite *paranoid and restrictive* with the definitions of environments and commands, the only thing left to do is do it with the ⟨*keys*⟩ (you have to be consistent in life).

12.35.1   Handling unknown keys for keyans and keyans*

unknown
__enumext_keyans_unknown_keys:n
__enumext_keyans_unknown_keys:nn

Define and set unknown key for keyans and keyans* environments.

```
3232  \cs_set_protected:Npn \__enumext_tmp:n #1
3233      {
3234        \keys_define:nn { enumext / #1 }
3235          {
3236            unknown .code:n = { \__enumext_keyans_unknown_keys:n {##1} }
3237          }
3238      }
3239  \clist_map_inline:nn { keyans, keyans* } { \__enumext_tmp:n {#1} }
```

Internal functions for handling unknown key.

```
3240  \cs_new_protected:Npn \__enumext_keyans_unknown_keys:n #1
3241      {
3242        \exp_args:NV \__enumext_keyans_unknown_keys:nn \l_keys_key_str {#1}
3243      }
3244  \cs_new_protected:Npn \__enumext_keyans_unknown_keys:nn #1#2
3245      {
3246        \tl_if_blank:nTF {#2}
3247          {
3248            \msg_error:nnn { enumext } { keyans-unknown-key } {#1}
3249          }
3250          {
3251            \msg_error:nnnn { enumext } { keyans-unknown-key-value } {#1} {#2}
3252          }
3253      }
```

(*End of definition for* unknown*,* __enumext_keyans_unknown_keys:n*, and* __enumext_keyans_unknown_keys:nn*.*)

12.35.2   Handling unknown keys for enumext*

unknown
__enumext_starred_unknown_keys:n
__enumext_starred_unknown_keys:nn

Define and set unknown key for enumext* environment.

```
3254  \keys_define:nn { enumext / enumext* }
3255      {
3256        unknown .code:n = { \__enumext_starred_unknown_keys:n {#1} }
3257      }
```

Internal functions for handling unknown key.

```
3258  \cs_new_protected:Npn \__enumext_starred_unknown_keys:n #1
3259      {
3260        \exp_args:NV \__enumext_starred_unknown_keys:nn \l_keys_key_str {#1}
3261      }
3262  \cs_new_protected:Npn \__enumext_starred_unknown_keys:nn #1#2
3263      {
3264        \tl_if_blank:nTF {#2}
3265          {
3266            \msg_error:nnn { enumext } { starred-unknown-key } {#1}
3267          }
3268          {
3269            \msg_error:nnnn { enumext } { starred-unknown-key-value } {#1} {#2}
3270          }
3271      }
```

(*End of definition for* unknown*,* __enumext_starred_unknown_keys:n*, and* __enumext_starred_unknown_keys:nn*.*)

12.35.3 Handling unknown keys for enumext

unknown

__enumext_standar_unknown_keys:n

__enumext_standar_unknown_keys:nn

Defines and set the key unknown for enumext environment.

```
3272 \cs_set_protected:Npn \__enumext_tmp:n #1
3273   {
3274     \keys_define:nn { enumext / #1 }
3275       {
3276         unknown .code:n = { \__enumext_standar_unknown_keys:n {##1} }
3277       }
3278   }
3279 \clist_map_inline:nn { level-1,level-2,level-3,level-4 } { \__enumext_tmp:n {#1} }
```

Internal functions for handling unknown key.

```
3280 \cs_new_protected:Npn \__enumext_standar_unknown_keys:n #1
3281   {
3282     \exp_args:NV \__enumext_standar_unknown_keys:nn \l_keys_key_str {#1}
3283   }
3284 \cs_new_protected:Npn \__enumext_standar_unknown_keys:nn #1#2
3285   {
3286     \tl_if_blank:nTF {#2}
3287       {
3288         \msg_error:nnn { enumext } { standar-unknown-key } {#1}
3289       }
3290       {
3291         \msg_error:nnnn { enumext } { standar-unknown-key-value } {#1} {#2}
3292       }
3293   }
```

(*End of definition for* unknown *,* __enumext_standar_unknown_keys:n *, and* __enumext_standar_unknown_keys:nn*.*)

12.36   Redefining \item and \makelabel in keyans

The \item and \item[⟨*custom*⟩] commands work in the usual way in keyans, but 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.

```
3294 \cs_new_protected:Npn \__enumext_keyans_default_item:n #1
3295   {
3296     \tl_if_novalue:nTF { #1 }
3297       {
3298         \bool_set_true:N \l__enumext_wrap_label_v_bool
3299         \__enumext_item_std:w \tl_use:N \l__enumext_fake_item_indent_v_tl
3300       }
3301       {
3302         \bool_set_eq:NN \l__enumext_wrap_label_v_bool \l__enumext_wrap_label_opt_v_bool
3303         \__enumext_item_std:w [#1] \tl_use:N \l__enumext_fake_item_indent_v_tl
3304       }
3305   }
```

(*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*⟩.

```
3306 \cs_new_protected:Npn \__enumext_keyans_starred_item:n #1
3307   {
3308     \tl_set_eq:NN \l__enumext_store_current_label_tmp_tl \l__enumext_label_v_tl
3309     \__enumext_keyans_show_left:n { #1 }
3310     \bool_set_true:N \l__enumext_wrap_label_v_bool
3311     \__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*⟩.

```
3312     \tl_set_eq:NN \l__enumext_label_v_tl \l__enumext_store_current_label_tmp_tl
3313     \__enumext_keyans_addto_prop:n { #1 }
3314     \__enumext_keyans_store_ref:
3315     \__enumext_keyans_addto_seq:n { #1 }
3316     \int_gincr:N \g__enumext_check_starred_cmd_int
3317   }
```

(*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.

```
3318 \cs_new_protected:Nn \__enumext_keyans_redefine_item:
3319   {
3320     \RenewDocumentCommand \item { s o }
3321       {
3322         \bool_if:nTF {##1}
3323           {
3324             \peek_remove_spaces:n
3325               {
3326                 \__enumext_keyans_starred_item:n {##2}
3327               }
3328           }
3329           {
3330             \__enumext_keyans_default_item:n {##2}
3331           }
3332       }
3333   }
```

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

__enumext_keyans_make_label:
__enumext_keyans_make_label_std:
__enumext_keyans_make_label_box:

The function __enumext_keyans_make_label: redefine \makelabel for the keys align, font, wrap-label, wrap-label* and \item* for keyans environment.

```
3334 \cs_new_protected:Nn \__enumext_keyans_make_label:
3335   {
3336     \IfDocumentMetadataTF
3337       {
3338         \__enumext_keyans_make_label_box:
3339       }
3340       { \__enumext_keyans_make_label_std: }
3341   }
3342 \cs_new_protected:Nn \__enumext_keyans_make_label_std:
3343   {
3344     \RenewDocumentCommand \makelabel { m }
3345       {
3346         \tl_use:N \l__enumext_label_fill_left_v_tl
3347         \tl_use:N \l__enumext_label_font_style_v_tl
3348         \bool_if:NTF \l__enumext_wrap_label_v_bool
3349           {
3350             \__enumext_wrapper_label_v:n { ##1 }
3351           }
3352           { ##1 }
3353         \tl_use:N \l__enumext_label_fill_right_v_tl
3354       }
3355   }
3356 \cs_new_protected:Nn \__enumext_keyans_make_label_box:
3357   {
3358     \RenewDocumentCommand \makelabel { m }
3359       {
3360         \makebox[ \l__enumext_labelwidth_v_dim ][ \l__enumext_align_label_pos_v_str ]
3361           {
3362             \tl_use:N \l__enumext_label_font_style_v_tl
3363             \bool_if:NTF \l__enumext_wrap_label_v_bool
3364               {
3365                 \__enumext_wrapper_label_v:n { ##1 }
3366               }
3367               { ##1 }
3368           }
3369       }
3370   }
```

(*End of definition for* __enumext_keyans_make_label:, __enumext_keyans_make_label_std:, *and* __enumext_keyans_-make_label_box:.)

🎯 This functions are passed to __enumext_list_arg_two_v: used in the definition of the keyans environment (§12.37.2).

12.37 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`.

12.37.1 Calculation of \leftmargin and \itemindent

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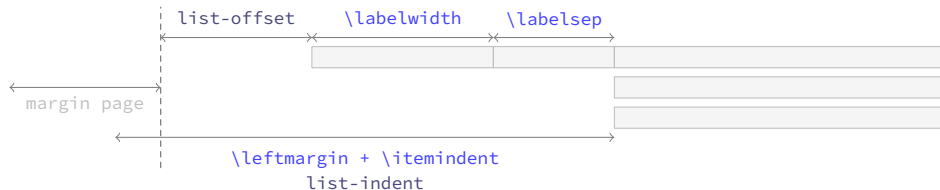
Figure 9: Representation of standard horizontal lengths in `list` environment.

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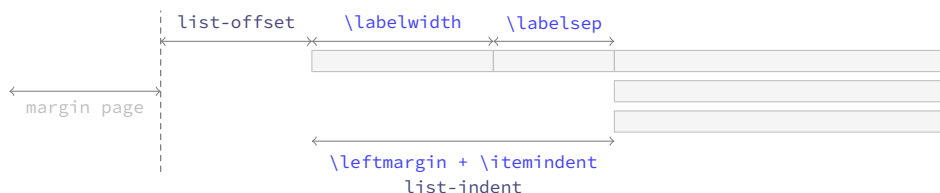
Figure 10: Representation of horizontal lengths concept in list in enumext.

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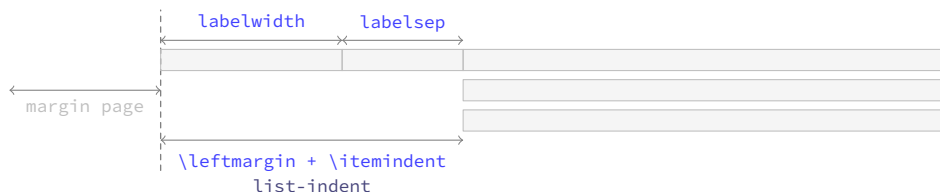
Figure 11: Default horizontal lengths in enumext.

__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 (§12.37.2).

```
3371  \cs_new_protected:Npn \__enumext_calc_hspace:NNNNNNN #1 #2 #3 #4 #5 #6 #7
3372    {
3373      \dim_compare:nNnT { #1 } < { \c_zero_dim }
3374        {
3375          \msg_warning:nnnV { enumext } { width-non-positive }{ labelwidth }{ #1 }
3376          \dim_set:Nn #1 { \dim_abs:n { #1 } }
3377        }
3378      \dim_compare:nNnT { #2 } < { \c_zero_dim }
3379        {
3380          \msg_warning:nnnV { enumext } { width-negative }{ labelsep }{ #2 }
3381          \dim_set:Nn #2 { \dim_abs:n { #2 } }
3382        }
```

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

```
3383      \bool_if:nF #7 { \dim_set:Nn #4 { #1 + #2} }
```

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

```
3384        \dim_compare:nNnTF { #4 } < { \c_zero_dim }
3385          {
3386            \dim_set:Nn #6 { #1 + #2 - #4}
3387            \dim_set:Nn #5 { #1 + #2 + #3 - #6 }
3388          }
3389          {
3390            \dim_compare:nNnT { #4 } = { #1 + #2 }
3391              { \dim_set:Nn #6 { \c_zero_dim } }
3392            \dim_compare:nNnT { #4 } < { #1 + #2 }
3393              { \dim_set:Nn #6 { #1 + #2 - #4} }
3394            \dim_compare:nNnT { #4 } > { #1 + #2 }
3395              {
3396                \dim_set:Nn #6 { -#1 - #2 + #4}
3397                \dim_set:Nn #6 { #6*-1}
3398              }
3399            \dim_set:Nn #5 { #1 + #2 + #3 - #6 }
3400          }
3401      }
3402  \cs_generate_variant:Nn \__enumext_calc_hspace:NNNNNNN { ccccccc }
```

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

12.37.2   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.

```
3403  \cs_set_protected:Npn \__enumext_tmp:n #1
3404    {
3405      \cs_new_protected:cpn { __enumext_list_arg_two_#1: }
3406        {
3407          \__enumext_calc_hspace:ccccccc
3408            { l__enumext_labelwidth_#1_dim } { l__enumext_labelsep_#1_dim }
3409            { l__enumext_listoffset_#1_dim } { l__enumext_leftmargin_tmp_#1_dim }
3410            { l__enumext_leftmargin_#1_dim } { l__enumext_itemindent_#1_dim }
3411            { l__enumext_leftmargin_tmp_#1_bool }
3412          \clist_map_inline:nn
3413            { labelsep, labelwidth, itemindent, leftmargin, rightmargin, listparindent }
3414            { \dim_set_eq:cc {####1} { l__enumext_####1_#1_dim } }
3415          \clist_map_inline:nn { topsep, parsep, partopsep, itemsep }
3416            { \skip_set_eq:cc {####1} { l__enumext_####1_#1_skip } }
3417          \usecounter { enumX#1 }
3418          \setcounter { enumX#1 } { \int_eval:n { \int_use:c { l__enumext_start_#1_int } - 1 } }
3419          \str_if_eq:nnTF {#1} { v }
3420            {
3421              \__enumext_keyans_redefine_item:
3422              \__enumext_keyans_make_label:
3423              \__enumext_keyans_ref:
3424              \__enumext_keyans_fake_item:
3425              \bool_if:cT { l__enumext_show_length_#1_bool }
3426                {
3427                  \msg_term:nnnn { enumext } { list-lengths-not-nested } { v } { keyans }
3428                }
3429            }
3430            {
3431              \__enumext_redefine_item:
3432              \__enumext_make_label:
3433              \__enumext_standar_ref:
3434              \__enumext_fake_item:
3435              \bool_if:cT { l__enumext_show_length_#1_bool }
3436                {
3437                  \msg_term:nnne { enumext } { list-lengths } {#1} { \int_use:N \l__enumext_level_i
3438                }
3439            }
3440        }
3441    }
3442  \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.

```
3443 \cs_set_protected:Npn \__enumext_tmp:n #1
3444   {
3445     \cs_new_protected:cpn { __enumext_list_arg_two_#1: }
3446       {
3447         \bool_set_true:c { l__enumext_leftmargin_tmp_#1_bool }
3448         \dim_zero:c { l__enumext_leftmargin_tmp_#1_dim }
3449         \__enumext_calc_hspace:ccccccc
3450           { l__enumext_labelwidth_#1_dim } { l__enumext_labelsep_#1_dim }
3451           { l__enumext_listoffset_#1_dim } { l__enumext_leftmargin_tmp_#1_dim }
3452           { l__enumext_leftmargin_#1_dim } { l__enumext_itemindent_#1_dim }
3453           { l__enumext_leftmargin_tmp_#1_bool }
3454         \clist_map_inline:nn
3455           { labelsep, labelwidth, itemindent, leftmargin, rightmargin, listparindent }
3456           { \dim_set_eq:cc {####1} { l__enumext_####1_#1_dim } }
3457         \clist_map_inline:nn { topsep, parsep, partopsep, itemsep }
3458           { \skip_set_eq:cc {####1} { l__enumext_####1_#1_skip } }
3459         \skip_set_eq:Nc \parsep  { l__enumext_itemsep_#1_skip }
3460         \skip_zero:N \partopsep
3461         \usecounter { enumX#1 }
3462         \setcounter { enumX#1 } { \int_eval:n { \int_use:c { l__enumext_start_#1_int } - 1 } }
3463         \__enumext_starred_ref:
3464         \str_if_eq:nnTF {#1} { vii }
3465           {
3466             \__enumext_fake_item_vii:
3467             \bool_if:cT { l__enumext_show_length_vii_bool }
3468               { \msg_term:nnnn { enumext } { list-lengths-not-nested } { vii } { enumext* } }
3469           }
3470           {
3471             \__enumext_fake_item_viii:
3472             \bool_if:cT { l__enumext_show_length_#1_bool }
3473               { \msg_term:nnnn { enumext } { list-lengths-not-nested } { #1 } { keyans* } }
3474           }
3475       }
3476   }
3477 \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:`.)

12.38 The environment enumext

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

```
3478 \NewDocumentEnvironment{enumext}{ O{} }
3479   {
3480     \__enumext_safe_exec:
3481     \__enumext_parse_keys:n {#1}
3482     \__enumext_before_list:
3483     \__enumext_start_store_level:
3484     \__enumext_start_list:nn
3485       { \tl_use:c { l__enumext_label_ \__enumext_level: _tl } }
3486       {
3487         \use:c { __enumext_list_arg_two_ \__enumext_level: : }
3488         \__enumext_before_keys_exec:
3489       }
3490     \__enumext_set_item_width:
3491     \__enumext_after_args_exec:
3492   }
3493   {
3494     \__enumext_after_list:
3495   }
```

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

__enumext_set_item_width: The function `\__enumext_set_item_width:` will set the value of `\itemwidth` taking into account the value established by the `list-offset` key for each level of the environment.

```
3496 \cs_new_protected:Nn \__enumext_set_item_width:
3497   {
3498     \dim_set:Nn \itemwidth
3499       {
3500         \linewidth
```

```
3501        }
3502      \dim_compare:nT
3503        {
3504          \dim_use:c { l__enumext_listoffset_ \__enumext_level: _dim } != \c_zero_dim
3505        }
3506        {
3507          \dim_sub:Nn \itemwidth
3508            {
3509              \dim_use:c { l__enumext_listoffset_ \__enumext_level: _dim }
3510            }
3511        }
3512      }
```

(*End of definition for* __enumext_set_item_width:.)

__enumext_safe_exec: The __enumext_safe_exec: function first call the function __enumext_internal_mini_page: to create the environment __enumext_mini_page, then the function __enumext_is_not_nested: which sets \g__enumext_standar_bool to *"true"* if we are not nested within enumext*, we will increment \l__enumext_level_int to restrict nesting of the environment, set \l__enumext_standar_bool to *"true"* and finally call the function __enumext_is_on_first_level: which sets \l__enumext_standar_first_bool to *"true"* only if the environment is not nested and we are at the *"first level"*.

```
3513  \cs_new_protected:Nn \__enumext_safe_exec:
3514    {
3515      \__enumext_internal_mini_page:
3516      \__enumext_is_not_nested:
3517      \int_incr:N \l__enumext_level_int
3518      \int_compare:nNnT { \l__enumext_level_int } > { 4 }
3519        { \msg_fatal:nn { enumext } { list-too-deep } }
3520      \bool_set_true:N \l__enumext_standar_bool
3521      \bool_set_false:N \l__enumext_starred_bool
3522      \__enumext_is_on_first_level:
3523    }
```

(*End of definition for* __enumext_safe_exec:.)

__enumext_parse_keys:n   The __enumext_parse_store_keys:n function first we will clear the variable \l__enumext_series_str used by the key series 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 and call the function __enumext_nested_base_line_fix: used by the key base-fix, otherwise we will pass the ⟨*keys*⟩ to the inner levels of the environment then we execute the function __enumext_store_active_keys:n and reprocess the ⟨*keys*⟩ to pass them to the storage ⟨*sequence*⟩ if the key save-key is not active.

```
3524  \cs_new_protected:Npn \__enumext_parse_keys:n #1
3525    {
3526      \tl_if_novalue:nF {#1}
3527        {
3528          \str_clear:N \l__enumext_series_str
3529          \int_compare:nNnTF { \l__enumext_level_int } = { 1 }
3530            {
3531              \keys_set:nn { enumext / level-1 } {#1}
3532              \__enumext_parse_series:n {#1}
3533              \__enumext_nested_base_line_fix:
3534            }
3535            {
3536              \exp_args:Ne \keys_set:nn
3537                { enumext / level-\int_use:N \l__enumext_level_int } {#1}
3538            }
3539          \__enumext_store_active_keys:n {#1}
3540        }
3541    }
```

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

__enumext_start_store_level: The __enumext_start_store_level: and __enumext_stop_store_level: functions activate the
__enumext_stop_store_level: level saving mechanism for storage in ⟨*sequence*⟩ for the command \anskey and the environment anskey*.

```
3542  \cs_new_protected:Nn \__enumext_start_store_level:
3543    {
3544      \bool_lazy_all:nT
3545        {
3546          { \bool_if_p:N \l__enumext_store_active_bool }
3547          { \bool_not_p:n { \l__enumext_keyans_env_bool } }
```

```
3548        { \bool_if_p:N \g__enumext_standar_bool }
3549      }
3550      {
3551        \int_compare:nNnT { \l__enumext_level_int } > { 1 }
3552          {
3553            \bool_set_true:c { l__enumext_store_upper_level_ \__enumext_level: _bool }
3554            \__enumext_store_level_open:
3555          }
3556      }
```

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

```
3557      \bool_lazy_all:nT
3558        {
3559          { \bool_if_p:N \l__enumext_store_active_bool }
3560          { \bool_not_p:n { \l__enumext_keyans_env_bool } }
3561          { \int_compare_p:nNn { \l__enumext_level_h_int } = { 1 } }
3562        }
3563        {
3564          \int_compare:nNnT { \l__enumext_level_int } > { 0 }
3565            {
3566              \bool_set_true:c { l__enumext_store_upper_level_ \__enumext_level: _bool }
3567              \__enumext_store_level_open:
3568            }
3569        }
3570    }
```

Close the stored structure.

```
3571  \cs_new_protected:Nn \__enumext_stop_store_level:
3572    {
3573      \bool_if:cT { l__enumext_store_upper_level_ \__enumext_level: _bool }
3574        {
3575          \__enumext_store_level_close:
3576        }
3577    }
```

(*End of definition for* __enumext_start_store_level: *and* __enumext_stop_store_level:*.*)

__enumext_before_list: The function __enumext_before_list: first calls the function __enumext_vspace_above: used by the keys above and above*, then calls the function __enumext_before_args_exec: used by the key before* and finally execute the function __enumext_check_ans_active: for the check answer mechanism.

```
3578  \cs_new_protected:Nn \__enumext_before_list:
3579    {
3580      \__enumext_vspace_above:
3581      \__enumext_before_args_exec:
3582      \__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_page 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_page environment on the *"left side"*, always having a current \linewidth as *maximum width* between them.

```
3583        \dim_compare:nNnT
3584          { \dim_use:c { l__enumext_minipage_right_ \__enumext_level: _dim } } > { \c_zero_dim }
3585          {
3586            \dim_set:cn { l__enumext_minipage_left_ \__enumext_level: _dim }
3587              {
3588                \linewidth
3589                - \dim_use:c { l__enumext_minipage_right_ \__enumext_level: _dim }
3590                - \dim_use:c { l__enumext_minipage_hsep_ \__enumext_level: _dim }
3591              }
```

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_minipage_add_space: is called and the __enumext_mini_page 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.

```
3592          \bool_set_true:c { l__enumext_minipage_active_ \__enumext_level: _bool }
3593          \int_gincr:N \g__enumext_minipage_stat_int
3594          \__enumext_minipage_add_space:
3595          \noindent
3596          \__enumext_mini_page{ \dim_use:c { l__enumext_minipage_left_ \__enumext_level: _dim } }
```

```
3597              }
3598          \__enumext_multicols_start:
3599      }
```

(*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.

```
3600  \cs_new_protected:Nn \__enumext_multicols_start:
3601      {
3602          \int_compare:nNnT
3603              { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } > { 1 }
3604              {
3605                  \dim_compare:nNnT
3606                      { \dim_use:c { l__enumext_columns_sep_ \__enumext_level: _dim } } = { \c_zero_dim }
3607                      {
3608                          \dim_set:cn { l__enumext_columns_sep_ \__enumext_level: _dim }
3609                              {
3610                                  ( \dim_use:c { l__enumext_labelwidth_ \__enumext_level: _dim }
3611                                    + \dim_use:c { l__enumext_labelsep_ \__enumext_level: _dim }
3612                                  ) / \int_use:c { l__enumext_columns_ \__enumext_level: _int }
3613                                  - \dim_use:c { l__enumext_listoffset_ \__enumext_level: _dim }
3614                              }
3615                      }
3616                  \dim_set_eq:Nc \columnsep { l__enumext_columns_sep_ \__enumext_level: _dim }
3617                  \int_compare:nNnT { \l__enumext_level_int } > { 1 }
3618                      {
3619                          \dim_zero:N \columnseprule
3620                      }
```

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.

```
3621                  \bool_if:cF { l__enumext_minipage_active_ \__enumext_level: _bool }
3622                      {
3623                          \skip_zero:N \multicolsep
3624                          \__enumext_multi_addvspace:
3625                      }
3626                  \raggedcolumns
3627                  \begin{multicols}{ \int_use:c { l__enumext_columns_ \__enumext_level: _int } }
3628              }
3629      }
```

(*End of definition for* __enumext_multicols_start:.)

__enumext_multicols_stop: The function __enumext_multicols_stop: will stop the multicols environment and apply our *"vertical adjust"* spacing.

```
3630  \cs_new_protected:Nn \__enumext_multicols_stop:
3631      {
3632          \int_compare:nNnTF
3633              { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } > { 1 }
3634              {
3635                  \__enumext_stop_list:
3636                  \__enumext_stop_store_level:
3637                  \end{multicols}
3638                  \__enumext_unskip_unkern:
3639                  \__enumext_unskip_unkern:
3640                  \par\addvspace{ \skip_use:c { l__enumext_multicols_below_ \__enumext_level: _skip } }
3641              }
3642              {
3643                  \__enumext_stop_list:
3644                  \__enumext_stop_store_level:
3645              }
3646      }
```

(*End of definition for* __enumext_multicols_stop:.)

__enumext_after_list: The function __enumext_after_list: first 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_page environment has not been closed), then close __enumext_-mini_page and add the *adjusted vertical space* \l__enumext_minipage_after_skip, otherwise we will close the multicols environment.

```
3647 \cs_new_protected:Nn \__enumext_after_list:
3648   {
3649     \bool_if:cTF { l__enumext_minipage_active_ \__enumext_level: _bool }
3650       {
3651         \int_compare:nNnT { \g__enumext_minipage_stat_int } = { 1 }
3652           {
3653             \msg_warning:nn { enumext } { missing-miniright }
3654             \miniright
3655           }
3656         \int_gzero:N \g__enumext_minipage_stat_int
3657         \__enumext_unskip_unkern: % remove topsep + [partopsep]
3658         \end__enumext_mini_page
3659       }
3660       {
3661         \__enumext_multicols_stop:
3662       }
```

Now we will execute the functions __enumext_after_stop_list: used by the key after, __enumext_-check_ans_key_hook: used by the key check-ans, __enumext_vspace_below: used by the keys below and below*. Finally set \l__enumext_standar_bool to false and call the function __enumext_resume_-save_counter: used by the series, resume and resume* keys.

```
3663     \__enumext_after_stop_list:
3664     \__enumext_check_ans_key_hook:
3665     \__enumext_vspace_below:
3666     \bool_set_false:N \l__enumext_standar_bool
3667     \__enumext_resume_save_counter:
3668   }
```

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.

```
3669 \__enumext_after_env:nn {enumext} { \__enumext_execute_after_env: }
```

(*End of definition for* __enumext_after_list:.)

12.39   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.

```
3670 \NewDocumentEnvironment{keyans}{ O{} }
3671   {
3672     \__enumext_keyans_safe_exec:
3673     \__enumext_keyans_parse_keys:n {#1}
3674     \__enumext_before_list_v:
3675     \__enumext_start_list:nn
3676       { \tl_use:N \l__enumext_label_v_tl }
3677       {
3678         \__enumext_list_arg_two_v:
3679         \__enumext_before_keys_exec_v:
3680       }
3681     \__enumext_keyans_set_item_width:
3682     \__enumext_after_args_exec_v:
3683   }
3684   {
3685     \__enumext_check_starred_cmd:n { item }
3686   %%\__enumext_stop_list:
3687     \__enumext_after_list_v:
3688   }
```

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

__enumext_keyans_set_item_width: The function __enumext_keyans_set_item_width: will set the value of \itemwidth taking into account the value established by the list-offset key.

```
3689 \cs_new_protected:Nn \__enumext_keyans_set_item_width:
3690   {
3691     \dim_set:Nn \itemwidth
3692       {
```

```
3693          \linewidth
3694        }
3695      \dim_compare:nT
3696        {
3697          \l__enumext_listoffset_v_dim != \c_zero_dim
3698        }
3699        {
3700          \dim_sub:Nn \itemwidth
3701            {
3702              \l__enumext_listoffset_v_dim
3703            }
3704        }
3705    }
```

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

`\__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.

```
3706  \cs_new_protected:Nn \__enumext_keyans_safe_exec:
3707    {
3708      \bool_if:NF \l__enumext_store_active_bool
3709        {
3710          \msg_error:nnnn { enumext } { wrong-place }{ keyans }{ save-ans }
3711        }
3712      \int_incr:N \l__enumext_keyans_level_int
3713      \bool_set_true:N \l__enumext_keyans_env_bool
3714      \__enumext_keyans_name_and_start:
3715      % Set false for interfering with enumext nested in keyans (yes, its possible and crayze)
3716      \bool_set_false:N \l__enumext_store_active_bool
3717      \int_compare:nNnT { \l__enumext_keyans_level_int } > { 1 }
3718        {
3719          \msg_error:nn { enumext } { keyans-nested }
3720        }
3721      \int_compare:nNnT { \l__enumext_level_int } > { 1 }
3722        {
3723          \msg_error:nn { enumext } { keyans-wrong-level }
3724        }
3725    }
```

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

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

```
3726  \cs_new_protected:Npn \__enumext_keyans_parse_keys:n #1
3727    {
3728      \keys_set:nn { enumext / keyans } {#1}
3729    }
```

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

`\__enumext_before_list_v:`
`\__enumext_keyans_multicols_start:`
`\__enumext_keyans_multicols_stop:`
`\__enumext_after_list_v:`

Same implementation as the one used in the `enumext` environment.

```
3730  \cs_new_protected:Nn \__enumext_before_list_v:
3731    {
3732      \__enumext_vspace_above_v:
3733      \__enumext_before_args_exec_v:
3734      \dim_compare:nNnT { \l__enumext_minipage_right_v_dim } > { \c_zero_dim }
3735        {
3736          \dim_set:Nn \l__enumext_minipage_left_v_dim
3737            {
3738              \linewidth - \l__enumext_minipage_right_v_dim - \l__enumext_minipage_hsep_v_dim
3739            }
3740          \bool_set_true:N \l__enumext_minipage_active_v_bool
3741          \int_gincr:N \g__enumext_minipage_stat_int
3742          \__enumext_keyans_minipage_add_space:
3743          \__enumext_mini_page{ \l__enumext_minipage_left_v_dim }
3744        }
3745      \__enumext_keyans_multicols_start:
3746    }
3747  \cs_new_protected:Nn \__enumext_keyans_multicols_start:
3748    {
3749      \int_compare:nNnT { \l__enumext_columns_v_int } > { 1 }
```

```
3750            {
3751              \dim_compare:nNnT { \l__enumext_columns_sep_v_dim } = { \c_zero_dim }
3752                {
3753                  \dim_set:Nn \l__enumext_columns_sep_v_dim
3754                    {
3755                      (
3756                        \l__enumext_labelwidth_v_dim + \l__enumext_labelsep_v_dim
3757                      ) / \l__enumext_columns_v_int
3758                      - \l__enumext_listoffset_v_dim
3759                    }
3760                }
3761              \dim_set_eq:NN \columnsep \l__enumext_columns_sep_v_dim
3762              \dim_zero:N \columnseprule % no rule here
3763              \bool_if:NF \l__enumext_minipage_active_v_bool
3764                {
3765                  \skip_zero:N \multicolsep
3766                  \__enumext_keyans_multi_addvspace:
3767                }
3768              \raggedcolumns
3769              \begin{multicols}{ \l__enumext_columns_v_int }
3770            }
3771      }
3772  \cs_new_protected:Nn \__enumext_keyans_multicols_stop:
3773      {
3774        \int_compare:nNnTF { \l__enumext_columns_v_int } > { 1 }
3775          {
3776            \__enumext_stop_list:
3777            \end{multicols}
3778            \__enumext_unskip_unkern:
3779            \__enumext_unskip_unkern:
3780            \par\addvspace{ \l__enumext_multicols_below_v_skip }
3781          }
3782          {
3783            \__enumext_stop_list:
3784          }
3785      }
3786  \cs_new_protected:Nn \__enumext_after_list_v:
3787      {
3788        \bool_if:NTF \l__enumext_minipage_active_v_bool
3789          {
3790            \int_compare:nNnT { \g__enumext_minipage_stat_int } = { 1 }
3791              {
3792                \msg_warning:nn { enumext } { missing-miniright }
3793                \miniright
3794              }
3795            \int_gzero:N \g__enumext_minipage_stat_int
3796            \__enumext_unskip_unkern: % remove topsep + [partopsep]
3797            \end__enumext_mini_page
3798            \par\addvspace{ \l__enumext_minipage_after_skip }
3799          }
3800          {
3801            \__enumext_keyans_multicols_stop:
3802          }
3803        \bool_set_false:N \l__enumext_keyans_env_bool
3804        \__enumext_after_stop_list_v:
3805        \__enumext_vspace_below_v:
3806      }
```

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

12.40 Tagging PDF support for non-standart list environments

The LaTeX release 2022-06-01 brings automatic support for `tagPDF` in several aspects, including the standart *list environments* and the `list` environment. Unfortunately non-standard *list environments* like `keyanspic` or the horizontal list environments `enumext*` and `keyans*` are not structured in a nice way, i.e. the expected result in the PDF file is the expected one, but the underlying structure is not correct. In simple terms, for `tagPDF` a list environment is a list environment, no matter what it looks like in the PDF file.

To maintain a correct list structure when `\DocumentMetadata` is active, it is necessary to do some things manually. This implementation is an adaptation of my answer thanks to Ulrike Fischer's comments in How can I modify my `\item` redefinition to be compatible with `tagging-pdf`.

12.40.1   Socket for tagging support in `enumext*` and `keyans*`

start-list-tags

stop-start-tags

stop-list-tags

__enumext_start_list_tag:n

__enumext_stop_start_list_tag:

__enumext_stop_list_tag:n

We will first define the necessary sockets and their behavior for `enumext*` and `keyans*`.

```
3807 \socket_new:nn {tagsupport/enumext/starred}{ 1 }
3808 \socket_new_plug:nnn {tagsupport/enumext/starred} {start-list-tags}
3809   {
3810     \tag_resume:n {#1}
3811     \tag_struct_begin:n {tag=LI}
3812     \tag_struct_begin:n {tag=Lbl}
3813     \tag_mc_begin:n {tag=Lbl}
3814   }
3815 \socket_new_plug:nnn {tagsupport/enumext/starred} {stop-start-tags}
3816   {
3817     \tag_mc_end:
3818     \tag_struct_end:n {tag=Lbl}
3819     \tag_struct_begin:n {tag=LBody}
3820     \tag_struct_begin:n {tag=text-unit}
3821     \tag_struct_begin:n {tag=text}
3822   }
3823 \socket_new_plug:nnn {tagsupport/enumext/starred} {stop-list-tags}
3824   {
3825     \tag_struct_end:n {tag=text}
3826     \tag_struct_end:n {tag=text-unit}
3827     \tag_struct_end:n {tag=LBody}
3828     \tag_struct_end:n {tag=LI}
3829     \tag_suspend:n {#1}
3830   }
```

And now we'll wrap them so that they're only active when `\DocumentMetadata` is present.

```
3831 \cs_new_protected_nopar:Npn \__enumext_start_list_tag:n #1
3832   {
3833     \IfDocumentMetadataTF
3834       {
3835         \socket_assign_plug:nn {tagsupport/enumext/starred} {start-list-tags}
3836         \socket_use:n {tagsupport/enumext/starred} {#1}
3837       } {}
3838   }
3839 \cs_new_protected_nopar:Nn \__enumext_stop_start_list_tag:
3840   {
3841     \IfDocumentMetadataTF
3842       {
3843         \socket_assign_plug:nn {tagsupport/enumext/starred} {stop-start-tags}
3844         \socket_use:nn {tagsupport/enumext/starred} { }
3845       } {}
3846   }
3847 \cs_new_protected_nopar:Npn \__enumext_stop_list_tag:n #1
3848   {
3849     \IfDocumentMetadataTF
3850       {
3851         \socket_assign_plug:nn {tagsupport/enumext/starred} {stop-list-tags}
3852         \socket_use:nn {tagsupport/enumext/starred} {#1}
3853       } {}
3854   }
```

(*End of definition for* start-list-tags *and others.*)

12.40.2   Socket for tagging support in `keyanspic`

start-list-tags

stop-start-tags

stop-list-tags

__enumext_anspic_start_list_tag:

__enumext_anspic_stop_start_list_tag:

__enumext_anspic_stop_list_tag:

We will first define the necessary sockets and their behavior for `keyanspic`.

```
3855 \socket_new:nn {tagsupport/enumext/keyanspic}{ 0 }
3856 \socket_new_plug:nnn {tagsupport/enumext/keyanspic} {start-list-tags}
3857   {
3858     \tag_resume:n {keyanspic}
3859     \tag_struct_begin:n {tag=LI}
3860     \tag_struct_begin:n {tag=Lbl}
3861     \tag_mc_begin:n {tag=Lbl}
3862   }
3863 \socket_new_plug:nnn {tagsupport/enumext/keyanspic} {stop-start-tags}
3864   {
3865     \tag_mc_end:
3866     \tag_struct_end:n {tag=Lbl}
3867     \tag_struct_begin:n {tag=LBody}
3868     \tag_struct_begin:n {tag=text-unit}
```

```
3869        \tag_struct_begin:n {tag=text}
3870        \tag_mc_begin:n {tag=text}
3871      }
3872  \socket_new_plug:nnn {tagsupport/enumext/keyanspic} {stop-list-tags}
3873      {
3874        \tag_mc_end:
3875        \tag_struct_end:n {tag=text-unit}
3876        \tag_struct_end:n {tag=text}
3877        \tag_struct_end:n {tag=LBody}
3878        \tag_struct_end:n {tag=LI}
3879        \tag_suspend:n {keyanspic}
3880      }
```

And now we'll wrap them so that they're only active when \DocumentMetadata is present.

```
3881  \cs_new_protected_nopar:Nn \__enumext_anspic_start_list_tag:
3882      {
3883        \IfDocumentMetadataTF
3884          {
3885            \socket_assign_plug:nn {tagsupport/enumext/keyanspic} {start-list-tags}
3886            \socket_use:n {tagsupport/enumext/keyanspic}
3887          } {}
3888      }
3889  \cs_new_protected_nopar:Nn \__enumext_anspic_stop_start_list_tag:
3890      {
3891        \IfDocumentMetadataTF
3892          {
3893            \socket_assign_plug:nn {tagsupport/enumext/keyanspic} {stop-start-tags}
3894            \socket_use:nn {tagsupport/enumext/keyanspic}
3895          } {}
3896      }
3897  \cs_new_protected_nopar:Nn \__enumext_anspic_stop_list_tag:
3898      {
3899        \IfDocumentMetadataTF
3900          {
3901            \socket_assign_plug:nn {tagsupport/enumext/keyanspic} {stop-list-tags}
3902            \socket_use:nn {tagsupport/enumext/keyanspic}
3903          } {}
3904      }
```

(*End of definition for* start-list-tags *and others.*)

12.41   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.
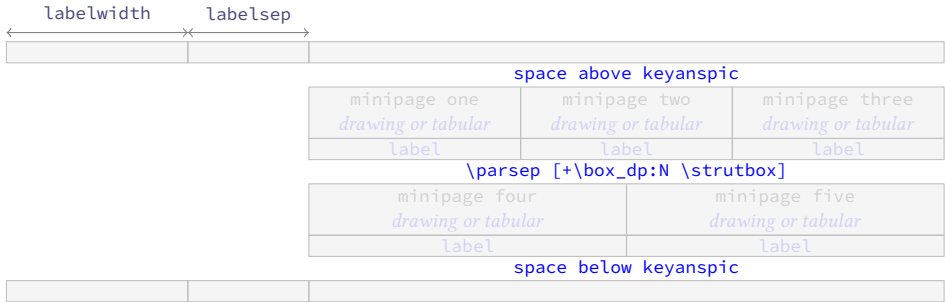
Figure 12: Representation of the keyanspic spacing in enumext.

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

12.41.1   The environment keyanspic

The environment we wish to build will be based on the list environment and will take two optional arguments, the *starred argument* ⋆ will set the position of the ⟨*label*⟩ to the top if present and bottom otherwise, the second optional argument will process the number of *drawing* or *tabular* that will be on top, bottom or inline when present or not.

__enumext_keyans_pic_safe_exec:n   The function __enumext_keyans_pic_safe_exec:n check the starred argument and nested level position inside the enumext environment. We will set the state of the variable \l__enumext_keyans_pic_star_-bool along with the value of the variable \l__enumext_keyans_pic_label_pos_str according to the presence of the starred argument.

```
3905  \cs_new_protected:Npn \__enumext_keyans_pic_safe_exec:n #1
3906    {
3907      \int_incr:N \l__enumext_keyans_pic_level_int
3908      \int_compare:nNnT { \l__enumext_keyans_pic_level_int } > { 1 }
3909        {
3910          \msg_error:nn { enumext } { keyanspic-nested }
3911        }
3912      \__enumext_keyans_name_and_start:
3913      \bool_if:nTF { #1 }
3914        {
3915          \bool_set_true:N \l__enumext_keyans_pic_star_bool
3916          \str_set:Nn \l__enumext_keyans_pic_label_pos_str { t }
3917        }
3918        {
3919          \str_set:Nn \l__enumext_keyans_pic_label_pos_str { b }
3920        }
3921    }
```

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

__enumext_keyans_pic_skip_abs:N   The function __enumext_keyans_pic_skip_abs:N will return a positive value \parsep.

```
3922  \cs_new_protected:Npn \__enumext_keyans_pic_skip_abs:N #1
3923    {
3924      \dim_compare:nNnT { #1 } < { \c_zero_dim }
3925        {
3926          \skip_set:Nn #1 { -#1 }
3927        }
3928    }
```

(*End of definition for* __enumext_keyans_pic_skip_abs:N.)

__enumext_keyans_pic_arg_two: The __enumext_keyans_pic_arg_two: function will be used in the second argument of the list environ-
ment that defines the keyanspic environment, with this we will take the configuration of the spaces and the
keys label and wrap-label from the keyans environment.
The first thing we need to do is set the boolean variable \l__enumext_leftmargin_tmp_v_bool handled
by the list-indent key to false, then copy the definition of the second list argument from the keyans
environment and make sure that \parsep does not have a negative value.

```
3929  \cs_new_protected:Nn \__enumext_keyans_pic_arg_two:
3930    {
3931      \bool_set_false:N \l__enumext_leftmargin_tmp_v_bool
3932      \__enumext_list_arg_two_v:
3933      \__enumext_keyans_pic_skip_abs:N \parsep
```

Now we increment the enumXv counter of the keyans environment and save the total height of our ⟨label⟩
in \l__enumext_anspic_label_htdp_dim and we will adjust the values of \parsep only if the starred
argument is NOT present (default version).

```
3934      \bool_if:NF \l__enumext_keyans_pic_star_bool
3935        {
3936          \stepcounter { enumXv }
3937          \hbox_set:Nn \l__enumext_anspic_label_box { \l__enumext_label_v_tl }
3938          \dim_set:Nn \l__enumext_anspic_label_htdp_dim
3939            {
3940              \box_ht_plus_dp:N \l__enumext_anspic_label_box
3941            }
3942          \skip_add:Nn \parsep
3943            {
3944              \l__enumext_anspic_label_htdp_dim + \box_dp:N \strutbox
3945            }
3946          \skip_gset_eq:NN \g__enumext_keyans_pic_parsep_skip \parsep
3947        }
```

Finally we adjust the value of \leftmargin and \topsep then set \labelwidth, \labelsep, \partopsep
and \itemsep to zero so that the horizontal and vertical space is not affected.

```
3948      \dim_add:Nn  \leftmargin { -\labelwidth - \labelsep }
3949      \skip_add:Nn \topsep { 0.5\box_dp:N \strutbox }
3950      \dim_zero:N  \labelwidth
3951      \dim_zero:N  \listparindent
3952      \dim_zero:N  \labelsep
3953      \skip_zero:N \partopsep
3954      \skip_zero:N \itemsep
3955    }
```

(End of definition for __enumext_keyans_pic_arg_two:.)

keyanspic   Now we define the environment keyanspic based on list. The starred argument will determine whether the ⟨*labels*⟩ will be displayed *"above"* or *"below"* the drawing or tabular. The second optional argument [⟨*number above, number below*⟩] will determine the number of minipage environments that will be above and below separated by \parsep within it.

```
3956  \NewDocumentEnvironment{keyanspic}{ s o }
3957    {
3958      \__enumext_keyans_pic_safe_exec:n { #1 }
3959      \begin{list} { } { \__enumext_keyans_pic_arg_two: }
3960      \IfDocumentMetadataTF
3961        {
3962          \tag_suspend:n {list}
3963        }{}
3964      \item[] \scan_stop:
3965      % paranoia
3966      \RenewDocumentCommand \item {}
3967        {
3968          \msg_error:nn { enumext } { keyanspic-item-cmd }
3969        }
3970      \IfDocumentMetadataTF
3971        {
3972          \tag_resume:n {keyanspic}
3973          \tag_tool:n {para/tagging=false}
3974          \tag_suspend:n {keyanspic}
3975        } { }
3976    }
```

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.

```
3977    {
3978      \IfDocumentMetadataTF
3979        {
3980          \tag_resume:n {keyanspic}
3981          \tag_struct_begin:n {tag=L,attribute=enumerate}
3982        } { }
3983      \tl_if_novalue:nTF { #2 }
3984        {
3985          \__enumext_keyans_pic_do:e { \seq_count:N \l__enumext_keyans_pic_body_seq }
3986        }
3987        { \__enumext_keyans_pic_do:n { #2 } }
3988      \IfDocumentMetadataTF
3989        {
3990          \tag_suspend:n {keyanspic}
3991        } { }
3992      \end{list}
3993      \IfDocumentMetadataTF { \tag_struct_end: \tag_struct_end: } { }
3994      \__enumext_check_starred_cmd:n { anspic }
3995      \setcounter { enumXvi } { 0 }
3996      \bool_if:NTF \l__enumext_keyans_pic_star_bool
3997        {
3998          \par\addvspace{ 0.5\box_dp:N \strutbox }
3999        }
4000        {
4001          \par\addvspace{ \g__enumext_keyans_pic_parsep_skip }
4002        }
4003      %\bool_set_false:N \l__enumext_store_active_bool
4004    }
```

(End of definition for keyanspic. This function is documented on page 15.)

__enumext_keyans_pic_do:n   The optional argument is split by comma and is handled directly by the function __enumext_keyans_-
__enumext_keyans_pic_do:e   pic_do:n and passed to the function __enumext_keyans_pic_row:n.

```
4005  \cs_new_protected:Nn \__enumext_keyans_pic_do:n
4006    {
4007      \clist_map_function:nN { #1 } \__enumext_keyans_pic_row:n
4008    }
4009  \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.

```
4010  \cs_new_protected:Nn \__enumext_keyans_pic_row:n
4011    {
4012      \dim_set:Nn \l__enumext_keyans_pic_width_dim { \linewidth / #1 }
4013      \int_set:Nn \l__enumext_keyans_pic_above_int { \l__enumext_keyans_pic_below_int }
4014      \int_set:Nn \l__enumext_keyans_pic_below_int { \l__enumext_keyans_pic_above_int + #1 }
4015      \int_step_inline:nnn
4016        { \l__enumext_keyans_pic_above_int + 1 }
4017        { \l__enumext_keyans_pic_below_int }
4018        {
4019          \IfDocumentMetadataTF
4020            {
4021              \tag_suspend:n {minipage}
4022            } { }
4023          \begin{minipage}[ \l__enumext_keyans_pic_label_pos_str ]{ \l__enumext_keyans_pic_width_dim
4024            \centering
4025            \seq_item:Nn \l__enumext_keyans_pic_body_seq { ##1 }
4026          \end{minipage}
4027          \IfDocumentMetadataTF
4028            {
4029              \tag_resume:n {minipage}
4030            } { }
4031        }
4032      \par
4033    }
```

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

12.41.2 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.

```
4034  \NewDocumentCommand \anspic { s o +m }
4035    {
```

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

```
4036      \bool_if:NF \l__enumext_store_active_bool
4037        {
4038          \msg_error:nnnn { enumext } { wrong-place }{ keyanspic }{ save-ans }
4039        }
4040      \int_compare:nNnT { \l__enumext_level_int } > { 1 }
4041        {
4042          \msg_error:nn { enumext } { keyanspic-wrong-level }
4043        }
4044      \int_compare:nNnT { \l__enumext_keyans_level_int } = { 1 }
4045        {
4046          \msg_error:nnnn { enumext } { command-wrong-place }{ anspic }{ keyans }
4047        }
```

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.

```
4048      \seq_put_right:Nn \l__enumext_keyans_pic_body_seq
4049        {
4050          \__enumext_keyans_anspic_code:nnn { #1 } { #2 } { #3 }
4051        }
4052    }
```

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

`\__enumext_anspic_box_set_dim:n`
`\__enumext_keyans_anspic_label:nnn`
`\__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.

```
4053  \cs_new_protected:Npn \__enumext_anspic_box_set_dim:n #1
4054    {
4055      \bool_if:NF \l__enumext_keyans_pic_star_bool
4056        {
4057          \IfDocumentMetadataTF
4058            {
```

```
4059                   \tag_suspend:n {keyanspic}
4060                 } { }
4061             \vbox_set:Nn \l__enumext_anspic_body_box { #1 }
4062             \dim_set:Nn \l__enumext_anspic_body_htdp_dim
4063               {
4064                 \box_ht_plus_dp:N \l__enumext_anspic_body_box
4065               }
4066             \IfDocumentMetadataTF
4067               {
4068                 \tag_resume:n {keyanspic}
4069               } { }
4070         }
4071     }
4072 % process label
4073 \cs_new_protected:Npn \__enumext_anspic_label:nn #1 #2
4074     {
4075       \makebox[ \l__enumext_keyans_pic_width_dim ][ c ]
4076         {
4077           \bool_if:nT { #1 }
4078             {
4079               \__enumext_keyans_addto_prop:n { #2 }
4080               \__enumext_keyans_store_ref:
4081               \__enumext_keyans_addto_seq:n { #2 }
4082               \int_gincr:N \g__enumext_check_starred_cmd_int
4083               \bool_lazy_or:nnT
4084                 { \bool_if_p:N \l__enumext_show_answer_bool }
4085                 { \bool_if_p:N \l__enumext_show_position_bool }
4086                 {
4087                   \tl_set_eq:NN \l__enumext_label_v_tl \l__enumext_label_vi_tl
4088                   \__enumext_keyans_show_left:n { #2 }
4089                   \tl_set_eq:NN \l__enumext_label_vi_tl \l__enumext_label_v_tl
4090                 }
4091             }
4092           \tl_use:N \l__enumext_label_font_style_v_tl
4093           \__enumext_wrapper_label_v:n { \l__enumext_label_vi_tl }
4094           \__enumext_keyans_show_item_opt:
4095         }
4096     }
4097 \cs_new_protected:Npn \__enumext_keyans_anspic_label:nnn #1 #2 #3
4098     {
4099       \stepcounter { enumXvi }
4100       \__enumext_anspic_box_set_dim:n { #3 }
4101       \bool_if:NTF \l__enumext_keyans_pic_star_bool
4102         {
4103           \__enumext_anspic_label:nn { #1 } { #2 }
4104         }
4105         {
4106           \raisebox
4107             {
4108               -\dim_eval:n
4109                 {
4110                   \l__enumext_anspic_label_htdp_dim
4111                   + \l__enumext_anspic_body_htdp_dim
4112                   + \box_dp:N \strutbox
4113                 }
4114             }
4115             [ 0pt ] [ 0pt ]
4116             {
4117               \__enumext_anspic_label:nn { #1 } { #2 }
4118             }
4119         }
4120     }
4121 \cs_new_protected:Nn \__enumext_keyans_anspic_code:nnn
4122     {
4123       \__enumext_anspic_start_list_tag:
4124       \__enumext_keyans_anspic_label:nnn { #1 } { #2 } { #3 }
4125       \__enumext_anspic_stop_start_list_tag:
4126       \\ #3
4127       \__enumext_anspic_stop_list_tag:
4128     }
```

(*End of definition for* __enumext_anspic_box_set_dim:n, __enumext_keyans_anspic_label:nnn, *and* __enumext_keyans_-

anspic_code:nnn.)

12.42   The horizontal environments

Generating *horizontal list environments* is NOT as simple as standard LaTeX list environments. The fundamental part of the code is adapted from the `shortlst` package to a more modern version using `expl3`. It is not possible to redefine `\item` and `\makelabel` as in the vertical *non starred* versions.

12.42.1   Functions for item box width

To achieve the horizontal list environment we will capture the `\item` command and the ⟨*content*⟩ of this in *horizontal 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*⟩).

We set the default value for the *width of the box* containing the ⟨*content*⟩ of the items for `enumext*` environment.

`\__enumext_starred_columns_set_vii:`
`\__enumext_starred_columns_set_viii:`

```
4129 \cs_new_protected:Nn \__enumext_starred_columns_set_vii:
4130   {
4131     \dim_compare:nNnT { \l__enumext_columns_sep_vii_dim } = { \c_zero_dim }
4132       {
4133         \dim_set:Nn \l__enumext_columns_sep_vii_dim
4134           {
4135             ( \l__enumext_labelwidth_vii_dim + \l__enumext_labelsep_vii_dim )
4136             / \l__enumext_columns_vii_int
4137           }
4138       }
4139     \int_set:Nn \l__enumext_tmpa_vii_int { \l__enumext_columns_vii_int - 1 }
4140     \dim_set:Nn \l__enumext_item_width_vii_dim
4141       {
4142         ( \linewidth - \l__enumext_columns_sep_vii_dim * \l__enumext_tmpa_vii_int )
4143         / \l__enumext_columns_vii_int
4144         - \l__enumext_labelwidth_vii_dim
4145         - \l__enumext_labelsep_vii_dim
4146       }
```

When the key `rightmargin` is active we must adjust the values.

```
4147     \dim_compare:nNnT { \l__enumext_rightmargin_vii_dim } > { \c_zero_dim }
4148       {
4149         \dim_sub:Nn \l__enumext_item_width_vii_dim
4150           {
4151             ( \l__enumext_rightmargin_vii_dim * \l__enumext_tmpa_vii_int )
4152             / \l__enumext_columns_vii_int
4153           }
4154         \dim_add:Nn \l__enumext_columns_sep_vii_dim
4155           {
4156             \l__enumext_rightmargin_vii_dim
4157           }
4158       }
4159   }
```

Same implementation for the `keyans*` environment.

```
4160 \cs_new_protected:Nn \__enumext_starred_columns_set_viii:
4161   {
4162     \dim_compare:nNnT { \l__enumext_columns_sep_viii_dim } = { \c_zero_dim }
4163       {
4164         \dim_set:Nn \l__enumext_columns_sep_viii_dim
4165           {
4166             ( \l__enumext_labelwidth_viii_dim + \l__enumext_labelsep_viii_dim )
4167             / \l__enumext_columns_viii_int
4168           }
4169       }
4170     \int_set:Nn \l__enumext_tmpa_viii_int { \l__enumext_columns_viii_int - 1 }
4171     \dim_set:Nn \l__enumext_item_width_viii_dim
4172       {
4173         ( \linewidth - \l__enumext_columns_sep_viii_dim * \l__enumext_tmpa_viii_int )
4174         / \l__enumext_columns_viii_int
4175         - \l__enumext_labelwidth_viii_dim
4176         - \l__enumext_labelsep_viii_dim
4177       }
4178     \dim_compare:nNnT { \l__enumext_rightmargin_viii_dim } > { \c_zero_dim }
4179       {
4180         \dim_sub:Nn \l__enumext_item_width_viii_dim
```

```
4181                {
4182                  ( \l__enumext_rightmargin_viii_dim * \l__enumext_tmpa_vii_int )
4183                  / \l__enumext_columns_viii_int
4184                }
4185            \dim_add:Nn \l__enumext_columns_sep_viii_dim
4186              {
4187                \l__enumext_rightmargin_viii_dim
4188              }
4189          }
4190      }
```

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

12.42.2 Functions for join item columns

__enumext_starred_joined_item_vii:n
__enumext_starred_joined_item_viii:n

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

```
4191  \cs_new_protected:Npn \__enumext_starred_joined_item_vii:n #1
4192    {
4193      \int_set:Nn \l__enumext_joined_item_vii_int {#1}
4194      \int_compare:nNnT { \l__enumext_joined_item_vii_int } > { \l__enumext_columns_vii_int }
4195        {
4196          \msg_warning:nnee { enumext } { item-joined }
4197            { \int_use:N \l__enumext_joined_item_vii_int }
4198            { \int_use:N \l__enumext_columns_vii_int }
4199          \int_set:Nn \l__enumext_joined_item_vii_int
4200            {
4201              \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + 1
4202            }
4203        }
4204      \int_compare:nNnT
4205        { \l__enumext_joined_item_vii_int }
4206          >
4207        { \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + 1 }
4208        {
4209          \msg_warning:nnee { enumext } { item-joined-columns }
4210            { \int_use:N \l__enumext_joined_item_vii_int }
4211            {
4212              \int_eval:n
4213                { \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + 1 }
4214            }
4215          \int_set:Nn \l__enumext_joined_item_vii_int
4216            {
4217              \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + 1
4218            }
4219        }
4220      \int_compare:nNnTF { \l__enumext_joined_item_vii_int } > { 1 }
4221        {
4222          \int_set_eq:NN \l__enumext_joined_item_aux_vii_int \l__enumext_joined_item_vii_int
4223          \int_decr:N \l__enumext_joined_item_aux_vii_int
4224          \int_add:Nn \l__enumext_item_column_pos_vii_int { \l__enumext_joined_item_aux_vii_int }
4225          \int_gadd:Nn \g__enumext_item_count_all_vii_int { \l__enumext_joined_item_aux_vii_int }
4226          \dim_set:Nn \l__enumext_joined_width_vii_dim
4227            {
4228              \l__enumext_item_width_vii_dim * \l__enumext_joined_item_vii_int
4229              + (  \l__enumext_labelwidth_vii_dim + \l__enumext_labelsep_vii_dim
4230                 + \l__enumext_columns_sep_vii_dim
4231                )*\l__enumext_joined_item_aux_vii_int
4232            }
4233          \dim_set_eq:NN \itemwidth \l__enumext_joined_width_vii_dim
4234        }
4235        {
4236          \dim_set_eq:NN \l__enumext_joined_width_vii_dim \l__enumext_item_width_vii_dim
4237          \dim_set_eq:NN \itemwidth \l__enumext_item_width_vii_dim
4238        }
4239    }
```

Same implementation for the keyans* environment.

```
4240  \cs_new_protected:Npn \__enumext_starred_joined_item_viii:n #1
4241    {
4242      \int_set:Nn \l__enumext_joined_item_viii_int {#1}
```

```
4243        \int_compare:nNnT { \l__enumext_joined_item_viii_int } > { \l__enumext_columns_viii_int }
4244          {
4245            \msg_warning:nnee { enumext } { item-joined }
4246              { \int_use:N \l__enumext_joined_item_viii_int }
4247              { \int_use:N \l__enumext_columns_viii_int }
4248            \int_set:Nn \l__enumext_joined_item_viii_int
4249              {
4250                \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1
4251              }
4252          }
4253        \int_compare:nNnT
4254          { \l__enumext_joined_item_viii_int }
4255            >
4256          { \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1 }
4257          {
4258            \msg_warning:nnee { enumext } { item-joined-columns }
4259              { \int_use:N \l__enumext_joined_item_viii_int }
4260              {
4261                \int_eval:n
4262                  { \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1 }
4263              }
4264            \int_set:Nn \l__enumext_joined_item_viii_int
4265              {
4266                \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1
4267              }
4268          }
4269        \int_compare:nNnTF { \l__enumext_joined_item_viii_int } > { 1 }
4270          {
4271            \int_set_eq:NN \l__enumext_joined_item_aux_viii_int \l__enumext_joined_item_viii_int
4272            \int_decr:N \l__enumext_joined_item_aux_viii_int
4273            \int_add:Nn \l__enumext_item_column_pos_viii_int { \l__enumext_joined_item_aux_viii_int }
4274            \int_gadd:Nn \g__enumext_item_count_all_viii_int { \l__enumext_joined_item_aux_viii_int }
4275            \dim_set:Nn \l__enumext_joined_width_viii_dim
4276              {
4277                \l__enumext_item_width_viii_dim * \l__enumext_joined_item_viii_int
4278                + ( \l__enumext_labelwidth_viii_dim + \l__enumext_labelsep_viii_dim
4279                    + \l__enumext_columns_sep_viii_dim
4280                  )*\l__enumext_joined_item_aux_viii_int
4281              }
4282            \dim_set_eq:NN \itemwidth \l__enumext_joined_width_viii_dim
4283          }
4284          {
4285            \dim_set_eq:NN \l__enumext_joined_width_viii_dim \l__enumext_item_width_viii_dim
4286            \dim_set_eq:NN \itemwidth \l__enumext_item_width_viii_dim
4287          }
4288      }
```

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

12.42.3 Functions for `mini-env`, `mini-right` and `mini-right*` keys

To have compatibility with the `tagPDF` we close the environment according to the presence or not of the `mini-right` key.

__enumext_start_mini_vii:
__enumext_stop_mini_vii:

The implementation of the `mini-env` key support is almost identical to the one used in the `enumext` and `keyans` environments, the difference is that the `__enumext_mini_page` 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.

```
4289  \cs_new_protected:Nn \__enumext_start_mini_vii:
4290    {
4291      \dim_compare:nNnT { \l__enumext_minipage_right_vii_dim } > { \c_zero_dim }
4292        {
4293          \dim_set:Nn \l__enumext_minipage_left_vii_dim
4294            {
4295              \linewidth
4296              - \l__enumext_minipage_right_vii_dim
4297              - \l__enumext_minipage_hsep_vii_dim
4298            }
4299          \bool_set_true:N \l__enumext_minipage_active_vii_bool
4300          \dim_gset_eq:NN
4301            \g__enumext_minipage_right_vii_dim
4302            \l__enumext_minipage_right_vii_dim
```

```
4303        \__enumext_mini_addvspace_vii:
4304        \nointerlineskip\noindent
4305        \__enumext_mini_page{ \l__enumext_minipage_left_vii_dim }
4306      }
4307    }
```

The function `\__enumext_stop_mini_vii:` closes the `__enumext_mini_page` environment on the *"left side"*, applies `\hfill` and set the variable `\g__enumext_minipage_active_vii_bool` to *"true"* which will be used in the function `\__enumext_after_env:nn` to execute the minipage on the *"right side"*.

```
4308  \cs_new_protected:Nn \__enumext_stop_mini_vii:
4309    {
4310      \bool_if:NTF \l__enumext_minipage_active_vii_bool
4311        {
4312          \__enumext_stop_list:
4313          \__enumext_stop_store_level_vii:
4314          \IfDocumentMetadataTF { \tag_resume:n {enumext*} } { }
4315          \end__enumext_mini_page
4316          \hfill
4317          \bool_gset_true:N \g__enumext_minipage_active_vii_bool
4318        }
4319        {
4320          \__enumext_stop_list:
4321          \__enumext_stop_store_level_vii:
4322        }
4323    }
```

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

```
4324  \__enumext_after_env:nn {enumext*}
4325    {
4326      \bool_if:NT \g__enumext_minipage_active_vii_bool
4327        {
4328          \__enumext_minipage:w [ t ] { \g__enumext_minipage_right_vii_dim }
4329            \legacy_if_gset_false:n { @minipage }
4330            \skip_vertical:N \c_zero_skip
4331            \par\addvspace { \g__enumext_minipage_right_skip }
4332            \bool_if:NF \g__enumext_minipage_center_vii_bool
4333              {
4334                \tl_put_left:Nn \g__enumext_miniright_code_vii_tl
4335                  {
4336                    \centering
4337                  }
4338              }
4339            \vbox_set_top:Nn \l__enumext_miniright_code_vii_box
4340              {
4341                \tl_use:N \g__enumext_miniright_code_vii_tl
4342              }
4343            \box_use_drop:N \l__enumext_miniright_code_vii_box
4344            \skip_vertical:N \c_zero_skip
4345          \__enumext_endminipage:
4346          \par\addvspace{ \g__enumext_minipage_after_skip }
4347        }
4348      \bool_gset_false:N \g__enumext_minipage_active_vii_bool
4349      \bool_gset_true:N \g__enumext_minipage_center_vii_bool
4350      \tl_gclear:N \g__enumext_miniright_code_vii_tl
4351      \dim_gzero:N \g__enumext_minipage_right_vii_dim
4352      \bool_gset_false:N \g__enumext_starred_bool
4353    }
```

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

`\__enumext_start_mini_viii:`
`\__enumext_stop_mini_viii:`

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

```
4354  \cs_new_protected:Nn \__enumext_start_mini_viii:
4355    {
4356      \dim_compare:nNnT { \l__enumext_minipage_right_viii_dim } > { \c_zero_dim }
4357        {
4358          \dim_set:Nn \l__enumext_minipage_left_viii_dim
4359            {
```

```
4360            \linewidth
4361              - \l__enumext_minipage_right_viii_dim
4362              - \l__enumext_minipage_hsep_viii_dim
4363            }
4364          \bool_set_true:N \l__enumext_minipage_active_viii_bool
4365          \dim_gset_eq:NN
4366            \g__enumext_minipage_right_viii_dim
4367            \l__enumext_minipage_right_viii_dim
4368          \__enumext_mini_addvspace_viii:
4369          \nointerlineskip\noindent
4370          \__enumext_mini_page{ \l__enumext_minipage_left_viii_dim }
4371        }
4372      }
4373  \cs_new_protected:Nn \__enumext_stop_mini_viii:
4374    {
4375      \bool_if:NTF \l__enumext_minipage_active_viii_bool
4376        {
4377          \__enumext_stop_list:
4378          \IfDocumentMetadataTF { \tag_resume:n {keyans*} } { }
4379          \end__enumext_mini_page
4380          \hfill
4381          \bool_gset_true:N \g__enumext_minipage_active_viii_bool
4382        }
4383        {
4384          \__enumext_stop_list:
4385        }
4386    }
4387  \__enumext_after_env:nn {keyans*}
4388    {
4389      \bool_if:NT \g__enumext_minipage_active_viii_bool
4390        {
4391          \__enumext_mini_page{ \g__enumext_minipage_right_viii_dim }
4392            \par\addvspace { \g__enumext_minipage_right_skip }
4393            \bool_if:NF \g__enumext_minipage_center_viii_bool
4394              {
4395                \tl_put_left:Nn \g__enumext_miniright_code_viii_tl
4396                  {
4397                    \centering
4398                  }
4399              }
4400            \vbox_set_top:Nn \l__enumext_miniright_code_viii_box
4401              {
4402                \tl_use:N \g__enumext_miniright_code_viii_tl
4403              }
4404            \box_use_drop:N \l__enumext_miniright_code_viii_box
4405          \end__enumext_mini_page
4406          \par\addvspace{ \g__enumext_minipage_after_skip }
4407        }
4408      \bool_gset_false:N \g__enumext_minipage_active_viii_bool
4409      \bool_gset_true:N \g__enumext_minipage_center_viii_bool
4410      \tl_gclear:N \g__enumext_miniright_code_viii_tl
4411      \dim_gzero:N \g__enumext_minipage_right_viii_dim
4412    }
```

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

12.42.4   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 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.

```
4413  \cs_new_protected:Nn \__enumext_footnotetext:nn
4414    {
4415      \footnotetext[#1]{#2}
4416    }
4417  \cs_new_protected:Nn \__enumext_renew_footnote:
4418    {
4419      \seq_gclear:N \g__enumext_footnote_arg_seq
4420      \seq_gclear:N \g__enumext_footnote_int_seq
4421      \RenewDocumentCommand \footnote { o +m }
4422        {
4423          \tl_if_novalue:nTF {##1}
```

```
4424            {
4425              \stepcounter{footnote}
4426              \int_gset_eq:Nc \g__enumext_footnote_int { c@footnote }
4427            }
4428            {
4429              \int_gset:Nn \g__enumext_footnote_int { ##1 }
4430            }
4431          \footnotemark [ \g__enumext_footnote_int ]
4432          \seq_gput_right:Nn \g__enumext_footnote_arg_seq { ##2 }
4433          \seq_gput_right:NV \g__enumext_footnote_int_seq \g__enumext_footnote_int
4434        }
4435      }
4436  \cs_new_protected:Nn \__enumext_print_footnote:
4437    {
4438      \seq_if_empty:NF \g__enumext_footnote_int_seq
4439        {
4440          \seq_map_pairwise_function:NNN
4441            \g__enumext_footnote_int_seq
4442            \g__enumext_footnote_arg_seq
4443            \__enumext_footnotetext:nn
4444        }
4445    }
```

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

12.43   The environment enumext*

enumext*   First we will generate the environment and we will give a temporary definition to __enumext_stop_-item_tmp_vii: equal to __enumext_first_item_tmp_vii: and next to \item equal to __enumext_-start_item_tmp_vii: which we will redefine later. Unlike the implementation used by the shortlst package, we will not set the values of \rightskip and \@rightskip equal to \@flushglue whose value is 0.0pt plus 1.0 fil, in the tests I have performed this fails in some circumstances and different results are obtained when using pdfTEX and LuaTEX.

```
4446  \NewDocumentEnvironment{enumext*}{ o }
4447    {
4448      \__enumext_safe_exec_vii:
4449      \__enumext_parse_keys_vii:n {#1}
4450      \__enumext_before_list_vii:
4451      \__enumext_start_store_level_vii:
4452      \__enumext_start_list:nn { }
4453        {
4454          \__enumext_list_arg_two_vii:
4455          \__enumext_before_keys_exec_vii:
4456        }
4457      % Stop tagging
4458      \IfDocumentMetadataTF { \tag_suspend:n {enumext*} } { }
4459      \__enumext_starred_columns_set_vii:
4460      \item[] \scan_stop:
4461      \cs_set_eq:NN \__enumext_stop_item_tmp_vii: \__enumext_first_item_tmp_vii:
4462      \cs_set_eq:NN \item \__enumext_start_item_tmp_vii:
4463      \ignorespaces
4464    }
4465    {
4466      \IfDocumentMetadataTF { \tag_struct_end:n {tag=text-unit} } { }
4467      \__enumext_stop_item_tmp_vii:
4468      \__enumext_remove_extra_parsep_vii:
4469      \__enumext_after_list_vii:
4470    }
```

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

__enumext_safe_exec_vii: We will first call the function __enumext_internal_mini_page: to create the environment __enumext_-mini_page, then the function __enumext_is_not_nested: which sets \g__enumext_starred_bool to true if we are not nested within enumext, we will increment \l__enumext_level_h_int to restrict nesting of the environment, set \l__enumext_starred_bool to true and finally call the function __enumext_is_-on_first_level: which sets \l__enumext_starred_first_bool to true if we are not nested, allowing the *"storage system"* to be used.

```
4471  \cs_new_protected:Nn \__enumext_safe_exec_vii:
4472    {
4473      \__enumext_internal_mini_page:
4474      \__enumext_is_not_nested:
```

```
4475        \int_incr:N \l__enumext_level_h_int
4476        \int_compare:nNnT { \l__enumext_level_h_int } > { 1 }
4477          {
4478            \msg_error:nn { enumext } { nested }
4479          }
4480        \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
4481          {
4482            \msg_error:nnn { enumext } { nested-horizontal } { keyans*}
4483          }
4484        \bool_set_true:N \l__enumext_starred_bool
4485        \bool_set_false:N \l__enumext_standar_bool
4486        \__enumext_is_on_first_level:
4487      }
```

(*End of definition for* __enumext_safe_exec_vii:.)

__enumext_parse_keys_vii:n   First we will clear the variable \l__enumext_series_str used by the key series, process the environment [⟨*key* = *val*⟩] and execute the function __enumext_parse_series:n and used by the key series, then we execute the function __enumext_store_active_keys_vii:n and reprocess the ⟨*keys*⟩ to pass them to the storage ⟨*sequence*⟩ if the key save-key is not active and finally we call the function __enumext_-nested_base_line_fix: used by the key base-fix.

```
4488  \cs_new_protected:Npn \__enumext_parse_keys_vii:n #1
4489    {
4490      \tl_if_novalue:nF {#1}
4491        {
4492          \str_clear:N \l__enumext_series_str
4493          \keys_set:nn { enumext / enumext* } {#1}
4494          \__enumext_parse_series:n {#1}
4495          \__enumext_store_active_keys_vii:n {#1}
4496          \__enumext_nested_base_line_fix:
4497        }
4498    }
```

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

__enumext_before_list_vii: The function __enumext_before_list_vii: first calls the function __enumext_vspace_above_vii: used by the keys above and above*, then calls the function __enumext_check_ans_active: for the check answer mechanism and finally calls the functions __enumext_before_args_exec: and __enumext_-start_mini_vii: used by the keys before*, mini-env, mini-right and mini-right*.

```
4499  \cs_new_protected:Nn \__enumext_before_list_vii:
4500    {
4501      \__enumext_vspace_above_vii:
4502      \__enumext_check_ans_active:
4503      \__enumext_before_args_exec_vii:
4504      \__enumext_start_mini_vii:
4505    }
```

(*End of definition for* __enumext_before_list_vii:.)

__enumext_after_list_vii: The function __enumext_after_list_vii: first calls the function __enumext_stop_mini_vii: used by the keys mini-env, mini-right and mini-right*, then to the functions __enumext_after_stop_-list_vii: used by the key after, __enumext_check_ans_key_hook: used by the key check-ans, __enumext_vspace_below_vii: used by the keys below and below*. Finally set \l__enumext_-starred_bool to false and call the __enumext_resume_save_counter: function used by the series, resume and resume* keys.

```
4506  \cs_new_protected:Nn \__enumext_after_list_vii:
4507    {
4508      \__enumext_stop_mini_vii:
4509      \__enumext_after_stop_list_vii:
4510      \__enumext_check_ans_key_hook:
4511      \__enumext_vspace_below_vii:
4512      \bool_set_false:N \l__enumext_starred_bool
4513      \__enumext_resume_save_counter:
4514    }
```

(*End of definition for* __enumext_after_list_vii:.)

__enumext_start_store_level_vii:
__enumext_stop_store_level_vii:

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

```
4515 \cs_new_protected:Nn \__enumext_start_store_level_vii:
4516   {
4517     \bool_if:NT \l__enumext_store_active_bool
4518       {
4519         \int_compare:nNnT { \l__enumext_level_int } > { 0 }
4520           {
4521             \__enumext_store_level_open_vii:
4522           }
4523       }
4524   }
4525 \cs_new_protected:Nn \__enumext_stop_store_level_vii:
4526   {
4527     \bool_if:NT \l__enumext_store_active_bool
4528       {
4529         \int_compare:nNnT { \l__enumext_level_int } > { 0 }
4530           {
4531             \__enumext_store_level_close_vii:
4532           }
4533       }
4534   }
```

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

12.43.1   The command \item in enumext*

__enumext_first_item_tmp_vii:

The __enumext_first_item_tmp_vii: function will remove horizontal space equal to \labelwidth plus \labelsep to the left of the first \item in the environment at the point of execution of this function, where it is equal to the __enumext_stop_item_tmp_vii: function inside the environment body definition.

```
4535 \cs_new_protected_nopar:Nn \__enumext_first_item_tmp_vii:
4536   {
4537     \skip_horizontal:n { -\l__enumext_labelwidth_vii_dim - \l__enumext_labelsep_vii_dim }
4538   }
```

(*End of definition for* __enumext_first_item_tmp_vii:.)

__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.

```
4539 \cs_new_protected_nopar:Nn \__enumext_start_item_tmp_vii:
4540   {
4541     \__enumext_stop_item_tmp_vii:
4542     \int_incr:N \l__enumext_item_column_pos_vii_int
4543     \int_gincr:N \g__enumext_item_count_all_vii_int
4544     \__enumext_item_peek_args_vii:
4545   }
```

(*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).

```
4546 \cs_new_protected:Nn \__enumext_item_peek_args_vii:
4547   {
4548     \peek_meaning:NTF (
4549       { \__enumext_joined_item_vii:w }
4550       { \__enumext_joined_item_vii:w (1) }
4551   }
```

(*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.

```
4552 \cs_new_protected:Npn \__enumext_joined_item_vii:w (#1)
4553   {
```

```
4554        \__enumext_starred_joined_item_vii:n {#1}
4555        \peek_meaning_remove:NTF *
4556          { \__enumext_starred_item_vii:w  }
4557          { \__enumext_standar_item_vii:w }
4558      }
```

(*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].

```
4559  \cs_new_protected:Npn \__enumext_standar_item_vii:w
4560    {
4561      \bool_set_false:N \l__enumext_item_starred_vii_bool
4562      \peek_meaning:NTF [
4563        {
4564          \bool_set_eq:NN \l__enumext_wrap_label_vii_bool \l__enumext_wrap_label_opt_vii_bool
4565          \__enumext_start_item_vii:w
4566        }
4567        {
4568          \bool_set_true:N \l__enumext_wrap_label_vii_bool
4569          \legacy_if_set_true:n { @noitemarg }
4570          \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
4571        }
4572    }
```

(*End of definition for* __enumext_standar_item_vii:w.)

__enumext_starred_item_vii:w   The function __enumext_starred_item_vii:w together with the specified auxiliary functions aux_i:w,
__enumext_starred_item_vii_aux_i:w   aux_ii:w, and aux_iii:w execute \item*, \item*[⟨*symbol*⟩] and \item*[⟨*symbol*⟩][⟨*offset*⟩].
__enumext_starred_item_vii_aux_ii:w
__enumext_starred_item_vii_aux_iii:w

```
4573  \cs_new_protected:Npn \__enumext_starred_item_vii:w
4574    {
4575      \bool_set_true:N \l__enumext_item_starred_vii_bool
4576      \bool_set_true:N \l__enumext_wrap_label_vii_bool
4577      \peek_meaning:NTF [
4578        { \__enumext_starred_item_vii_aux_i:w }
4579        { \__enumext_starred_item_vii_aux_ii:w }
4580    }
4581  \cs_new_protected:Npn \__enumext_starred_item_vii_aux_i:w [#1]
4582    {
4583      \tl_gset:Nn \g__enumext_item_symbol_aux_vii_tl {#1}
4584      \__enumext_starred_item_vii_aux_ii:w
4585    }
4586  \cs_new_protected:Npn \__enumext_starred_item_vii_aux_ii:w
4587    {
4588      \peek_meaning:NTF [
4589        { \__enumext_starred_item_vii_aux_iii:w }
4590        {
4591          \dim_set_eq:NN \l__enumext_item_symbol_sep_vii_dim \l__enumext_labelsep_vii_dim
4592          \legacy_if_set_true:n { @noitemarg }
4593          \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
4594        }
4595    }
4596  \cs_new_protected:Npn \__enumext_starred_item_vii_aux_iii:w [#1]
4597    {
4598      \dim_set:Nn \l__enumext_item_symbol_sep_vii_dim {#1}
4599      \legacy_if_set_true:n { @noitemarg }
4600      \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
4601    }
```

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

__enumext_fake_make_label_vii:n   The __enumext_fake_make_label_vii:n function will be in charge of handling our definition of \item. First we increment the counter enumXvii for the enumerated items and activate support for the *check answers* mechanism, followed by support for \item*[⟨*symbol*⟩][⟨*offset*⟩] if present, then the wrap-label and wrap-label* keys which we execute using \makebox whose width will be given by the labelwidth key and

position by the `align` key, inside the argument of this we will execute the `font` key together with the function defined by the `wrap-label` or `wrap-label*` keys. Finally we execute the `labelsep` key applying a *horizontal space*.

```
4602  \cs_new_protected_nopar:Npn \__enumext_fake_make_label_vii:n #1
4603    {
4604      \legacy_if:nT { @noitemarg }
4605        {
4606          \legacy_if_set_false:n { @noitemarg }
4607          \legacy_if:nT { @nmbrlist }
4608            {
4609              \refstepcounter{enumXvii}
4610              \bool_if:NT \l__enumext_check_answers_bool
4611                {
4612                  \int_gincr:N \g__enumext_item_number_int
4613                  \bool_set_true:N \l__enumext_item_number_bool
4614                }
4615            }
4616        }
4617      \bool_if:NT \l__enumext_item_starred_vii_bool
4618        {
4619          \tl_if_blank:VT \g__enumext_item_symbol_aux_vii_tl
4620            {
4621              \tl_gset_eq:NN
4622                \g__enumext_item_symbol_aux_vii_tl \l__enumext_item_symbol_vii_tl
4623            }
4624          \mode_leave_vertical:
4625          \skip_horizontal:n { -\l__enumext_item_symbol_sep_vii_dim }
4626          \hbox_overlap_left:n { \g__enumext_item_symbol_aux_vii_tl }
4627          \skip_horizontal:N \l__enumext_item_symbol_sep_vii_dim
4628          \tl_gclear:N \g__enumext_item_symbol_aux_vii_tl
4629        }
4630      \makebox[ \l__enumext_labelwidth_vii_dim ][ \l__enumext_align_label_vii_str ]
4631        {
4632          \tl_use:N \l__enumext_label_font_style_vii_tl
4633          \bool_if:NTF \l__enumext_wrap_label_vii_bool
4634            {
4635              \__enumext_wrapper_label_vii:n {#1}
4636            }
4637            { #1 }
4638        }
4639      \skip_horizontal:N \l__enumext_labelsep_vii_dim
4640    }
```

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

12.43.2 Real definition of `\item` in enumext*

The functions `\__enumext_start_item_vii:w` and `\__enumext_stop_item_vii:` executing the true definition of `\item` inside the `enumext*` environment, unlike the implementation in `shortlst` we will NOT use an extra group and the plain form of the `lrbox` environment.

`\__enumext_start_item_vii:w` 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, after that we will start capturing `\item` and its ⟨*contents*⟩ in a *horizontal box* where the width will be `\itemwidth` plus `\labelwidth` plus `\labelsep`.

```
4641  \cs_new_protected_nopar:Npn \__enumext_start_item_vii:w [#1]
4642    {
4643      \cs_set_eq:NN \__enumext_stop_item_tmp_vii: \__enumext_stop_item_vii:
4644      \hbox_set_to_wd:Nnw \l__enumext_item_text_vii_box
4645        {
4646          \l__enumext_joined_width_vii_dim
4647          + \l__enumext_labelwidth_vii_dim
4648          + \l__enumext_labelsep_vii_dim
4649        }
```

If `\DocumentMetadata` is not active and the state of the variable `\l__enumext_footnotes_key_bool` is false, we will redefine the `\footnote` command.

```
4650      \IfDocumentMetadataTF { }
4651        {
4652          \bool_if:NF \l__enumext_footnotes_key_bool
4653            {
4654              \__enumext_renew_footnote:
```

4655   }
4656   }

Now we insert our *sockets* for the tagPDF support and print \item.

4657   __enumext_start_list_tag:n {enumext*}
4658   __enumext_fake_make_label_vii:n {#1}
4659   __enumext_stop_start_list_tag:

Finally we open the minipage environment capture the ⟨*item content*⟩ and execute the first key, listparindent key which will be equal to \parindent, the parsep key which will be equal to \parskip and the itemindent key.

4660   __enumext_minipage:w [t]{ \l__enumext_joined_width_vii_dim }
4661   \tl_use:N \l__enumext_after_list_args_vii_tl
4662   \dim_set_eq:NN \parindent \l__enumext_listparindent_vii_dim
4663   \skip_set_eq:NN \parskip \l__enumext_parsep_vii_skip
4664   \tl_use:N \l__enumext_fake_item_indent_vii_tl
4665   }

(*End of definition for* __enumext_start_item_vii:w.)

__enumext_stop_item_vii: The __enumext_stop_item_vii: function will finish the fetching \item and its ⟨*content*⟩ by closing the minipage environment, the *sockets* for the tagPDF and the *horizontal box*.

4666 \cs_new_protected_nopar:Nn __enumext_stop_item_vii:
4667   {
4668   __enumext_endminipage:
4669   __enumext_stop_list_tag:n {enumext*}
4670   \hbox_set_end:

Here we will reduce the *warnings* a bit by setting the value of \hbadness to 10000, print the ⟨*contents*⟩ of the *box* along with \footnote.

4671   \int_set:Nn \hbadness { 10000 }
4672   \box_use_drop:N \l__enumext_item_text_vii_box
4673   \IfDocumentMetadataTF { }
4674   {
4675   \bool_if:NF \l__enumext_footnotes_key_bool
4676   {
4677   __enumext_print_footnote:
4678   }
4679   }

Finally set the *vertical* and *horizontal* spaces between rows and columns.

4680   \int_compare:nNnTF
4681   { \l__enumext_item_column_pos_vii_int } = { \l__enumext_columns_vii_int }
4682   {
4683   \par\noindent
4684   \int_zero:N \l__enumext_item_column_pos_vii_int
4685   }
4686   {
4687   \skip_horizontal:N \l__enumext_columns_sep_vii_dim
4688   }
4689   }

(*End of definition for* __enumext_stop_item_vii:.)

__enumext_remove_extra_parsep_vii: Finally we will remove the vertical space equal to \parsep=\itemsep when the total number of items is divisible by the number of items in the last row of the environment. Here the use of \unskip or \removelastskip fails and does not obtain the expected result, using \vspace is the option and in this case, we can use a simplified version since we are always in ⟨*vertical mode*⟩.

4690 \cs_new_protected:Nn __enumext_remove_extra_parsep_vii:
4691   {
4692   \int_compare:nNnT
4693   {
4694   \int_mod:nn
4695   { \g__enumext_item_count_all_vii_int } { \l__enumext_columns_vii_int }
4696   }
4697   =
4698   { 0 }
4699   {
4700   \para_end:
4701   \skip_vertical:n { -\l__enumext_itemsep_vii_skip }
4702   \skip_vertical:N \c_zero_skip
4703   \int_gzero:N \g__enumext_item_count_all_vii_int

```
4704        }
4705    }
```

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`.

```
4706  \__enumext_after_env:nn {enumext*} { \__enumext_execute_after_env: }
```

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

12.44   The environment keyans*

keyans* First we will generate the environment and we will give a temporary definition to `\__enumext_stop_item_-tmp_viii:` equal to `\__enumext_first_item_tmp_viii:` and next to `\item` equal to `\__enumext_-start_item_tmp_viii:` which we will redefine later. The implementation of this environment is the same as that used by the `enumext*` environment except for the `\__enumext_check_starred_cmd:n` function added in the second part.

```
4707  \NewDocumentEnvironment{keyans*}{ o }
4708    {
4709      \__enumext_safe_exec_viii:
4710      \__enumext_parse_keys_viii:n {#1}
4711      \__enumext_before_list_viii:
4712      \__enumext_start_list:nn { }
4713        {
4714          \__enumext_list_arg_two_viii:
4715          \__enumext_before_keys_exec_viii:
4716        }
4717      \IfDocumentMetadataTF { \tag_suspend:n {keyans*} } { }
4718      \__enumext_starred_columns_set_viii:
4719      \item[] \scan_stop:
4720      \cs_set_eq:NN \__enumext_stop_item_tmp_viii: \__enumext_first_item_tmp_viii:
4721      \cs_set_eq:NN \item \__enumext_start_item_tmp_viii:
4722      \ignorespaces
4723    }
4724    {
4725      \IfDocumentMetadataTF { \tag_struct_end: } { }
4726      \__enumext_stop_item_tmp_viii:
4727      \__enumext_remove_extra_parsep_viii:
4728      \__enumext_check_starred_cmd:n { item }
4729      \__enumext_after_list_viii:
4730    }
```

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

__enumext_safe_exec_viii: The `\__enumext_safe_exec_viii:` function will first check if the `save-ans` key is active and only when this is true the environment will be available, it will increment the value of `\l__enumext_keyans_level_h_int` and return an error message when we are nesting the environment, then it will call the `\__enumext_-keyans_name_and_start:` function in charge of saving the name of the environment and the line it is running on, then it will check if we are trying to nest `keyans*` in `enumext*` returning an error and we will set `\l__enumext_starred_bool` to true, finally we will check if we are within the appropriate level within the `enumext` environment.

```
4731  \cs_new_protected:Nn \__enumext_safe_exec_viii:
4732    {
4733      \bool_if:NF \l__enumext_store_active_bool
4734        {
4735          \msg_error:nnnn { enumext } { wrong-place }{ keyans* }{ save-ans }
4736        }
4737      \int_incr:N \l__enumext_keyans_level_h_int
4738      \int_compare:nNnT { \l__enumext_keyans_level_h_int } > { 1 }
4739        {
4740          \msg_error:nn { enumext } { nested }
4741        }
4742      \__enumext_keyans_name_and_start:
4743      \bool_if:NT \l__enumext_starred_bool
4744        {
4745          \msg_error:nnn { enumext } { nested-horizontal } { enumext* }
4746        }
4747      \bool_set_true:N \l__enumext_starred_bool
4748      % Set false for interfering with enumext nested in keyans* (yes, its possible and crayze)
4749      \bool_set_false:N \l__enumext_store_active_bool
4750      \int_compare:nNnT { \l__enumext_level_int } > { 1 }
4751        {
```

```
4752          \msg_error:nn { enumext } { keyans-wrong-level }
4753        }
4754    }
```

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

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

```
4755 \cs_new_protected:Npn \__enumext_parse_keys_viii:n #1
4756    {
4757      \tl_if_novalue:nF {#1}
4758        {
4759          \keys_set:nn { enumext / keyans* } {#1}
4760        }
4761    }
```

(*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.

```
4762 \cs_new_protected:Nn \__enumext_before_list_viii:
4763    {
4764      \__enumext_vspace_above_viii:
4765      \__enumext_before_args_exec_viii:
4766      \__enumext_start_mini_viii:
4767    }
```

(*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.

```
4768 \cs_new_protected:Nn \__enumext_after_list_viii:
4769    {
4770      \__enumext_stop_mini_viii:
4771      \__enumext_after_stop_list_viii:
4772      \__enumext_vspace_below_viii:
4773    }
```

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

12.44.1   The command \item in keyans*

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

__enumext_first_item_tmp_viii: The __enumext_first_item_tmp_viii: function will remove horizontal space equal to \labelwidth plus \labelsep to the left of the first \item in the environment at the point of execution of this function, where it is equal to the __enumext_stop_item_tmp_viii: function inside the environment body definition.

```
4774 \cs_new_protected_nopar:Nn \__enumext_first_item_tmp_viii:
4775    {
4776      \skip_horizontal:n { -\l__enumext_labelwidth_viii_dim - \l__enumext_labelsep_viii_dim }
4777    }
```

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

__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.

```
4778 \cs_new_protected_nopar:Nn \__enumext_start_item_tmp_viii:
4779    {
4780      \__enumext_stop_item_tmp_viii:
4781      \int_incr:N \l__enumext_item_column_pos_viii_int
4782      \int_gincr:N \g__enumext_item_count_all_viii_int
4783      \__enumext_item_peek_args_viii:
4784    }
```

(*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).

```
4785 \cs_new_protected:Nn \__enumext_item_peek_args_viii:
4786   {
4787     \peek_meaning:NTF (
4788       { \__enumext_joined_item_viii:w }
4789       { \__enumext_joined_item_viii:w (1) }
4790   }
```

(*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.

```
4791 \cs_new_protected:Npn \__enumext_joined_item_viii:w (#1)
4792   {
4793     \__enumext_starred_joined_item_viii:n {#1}
4794     \peek_meaning_remove:NTF *
4795       { \__enumext_starred_item_viii:w  }
4796       { \__enumext_standar_item_viii:w }
4797   }
```

(*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].

```
4798 \cs_new_protected:Npn \__enumext_standar_item_viii:w
4799   {
4800     \bool_set_false:N \l__enumext_item_starred_viii_bool
4801     \peek_meaning:NTF [
4802       {
4803         \bool_set_eq:NN \l__enumext_wrap_label_viii_bool \l__enumext_wrap_label_opt_viii_bool
4804         \__enumext_start_item_viii:w
4805       }
4806       {
4807         \bool_set_true:N \l__enumext_wrap_label_viii_bool
4808         \legacy_if_set_true:n { @noitemarg }
4809         \__enumext_start_item_viii:w [ \l__enumext_label_viii_tl ]
4810       }
4811   }
```

(*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*⟩].

```
4812 \cs_new_protected:Npn \__enumext_starred_item_viii:w
4813   {
4814     \bool_set_true:N \l__enumext_item_starred_viii_bool
4815     \bool_set_true:N \l__enumext_wrap_label_viii_bool
4816     \peek_meaning:NTF [
4817       { \__enumext_starred_item_viii_aux_i:w }
4818       { \__enumext_starred_item_viii_aux_ii:w }
4819   }
```

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.

```
4820 \cs_new_protected:Npn \__enumext_starred_item_viii_aux_i:w [#1]
4821   {
4822     \tl_clear:N \l__enumext_store_current_label_tl
```

```
4823      \tl_if_novalue:nF { #1 }
4824        {
4825          \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_tl
4826            {
4827              \tl_put_right:Ne \l__enumext_store_current_label_tl
4828                {
4829                  \l__enumext_store_keyans_item_opt_sep_tl
4830                }
4831              \tl_put_right:Ne \l__enumext_store_current_label_tl { #1 }
4832            }
4833          \tl_set:Ne \l__enumext_store_current_opt_arg_tl { #1 }
4834        }
4835      \__enumext_starred_item_viii_aux_ii:w
4836    }
4837 \cs_new_protected:Npn \__enumext_starred_item_viii_aux_ii:w
4838    {
4839      \legacy_if_set_true:n { @noitemarg }
4840      \__enumext_start_item_viii:w [ \l__enumext_label_viii_tl ]
4841    }
```

(*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.

```
4842 \cs_new_protected:Nn \__enumext_starred_item_exec:
4843    {
4844      \tl_put_left:Ne \l__enumext_store_current_label_tl { \l__enumext_label_viii_tl }
4845      \__enumext_store_addto_prop:V \l__enumext_store_current_label_tl
4846      \__enumext_keyans_store_ref:
4847      \tl_put_left:Ne \l__enumext_store_current_label_tl { \item }
4848      \__enumext_keyans_addto_seq_link:
4849      \int_gincr:N \g__enumext_check_starred_cmd_int
4850      \bool_if:NT \l__enumext_show_answer_bool
4851        {
4852          \__enumext_print_keyans_box:NN \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
4853        }
4854      \bool_if:NT \l__enumext_show_position_bool
4855        {
4856          \tl_set:Ne \l__enumext_mark_answer_sym_tl
4857            {
4858              \group_begin:
4859                \exp_not:N \normalfont
4860                \exp_not:N \footnotesize [ \int_eval:n
4861                  {
4862                    \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
4863                  }
4864                ]
4865              \group_end:
4866            }
4867          \__enumext_print_keyans_box:NN \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
4868        }
4869    }
```

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

`\__enumext_fake_make_label_viii:n` The implementation at this is very similar to that of the enumext* environment.

```
4870 \cs_new_protected_nopar:Npn \__enumext_fake_make_label_viii:n #1
4871    {
4872      \legacy_if:nT { @noitemarg }
4873        {
4874          \legacy_if_set_false:n { @noitemarg }
4875          \legacy_if:nT { @nmbrlist }
4876            {
4877              \refstepcounter{enumXviii}
4878            }
4879        }
4880      \bool_if:NT \l__enumext_item_starred_viii_bool
4881        {
4882          \__enumext_starred_item_exec:
```

```
4883          }
4884       \makebox[ \l__enumext_labelwidth_viii_dim ][ \l__enumext_align_label_viii_str ]
4885         {
4886            \tl_use:N \l__enumext_label_font_style_viii_tl
4887            \bool_if:NTF \l__enumext_wrap_label_viii_bool
4888              {
4889                 \__enumext_wrapper_label_viii:n {#1}
4890              }
4891              { #1 }
4892         }
4893       \skip_horizontal:N \l__enumext_labelsep_viii_dim
4894     }
```

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

12.44.2 Real definition of \item in keyans*

__enumext_start_item_viii:w   The implementation at this is very similar to that of the enumext* environment.

```
4895 \cs_new_protected_nopar:Npn \__enumext_start_item_viii:w [#1]
4896    {
4897       \cs_set_eq:NN \__enumext_stop_item_tmp_viii: \__enumext_stop_item_viii:
4898       \hbox_set_to_wd:Nnw \l__enumext_item_text_viii_box
4899         {
4900            \l__enumext_joined_width_viii_dim
4901            + \l__enumext_labelwidth_viii_dim
4902            + \l__enumext_labelsep_viii_dim
4903         }
4904       \IfDocumentMetadataTF { }
4905         {
4906            \bool_if:NF \l__enumext_footnotes_key_bool
4907              {
4908                 \__enumext_renew_footnote:
4909              }
4910         }
4911       \__enumext_start_list_tag:n {keyans*}
4912       \__enumext_fake_make_label_viii:n {#1}
4913       \__enumext_stop_start_list_tag:
4914       \__enumext_minipage:w [ t ]{ \l__enumext_joined_width_viii_dim  }
4915         \tl_use:N \l__enumext_after_list_args_viii_tl
4916         \dim_set_eq:NN \parindent \l__enumext_listparindent_viii_dim
4917         \skip_set_eq:NN \parskip \l__enumext_parsep_viii_skip
4918         \bool_if:NT \l__enumext_item_starred_viii_bool
4919           {
4920              \tl_use:N \l__enumext_fake_item_indent_viii_tl
4921              \__enumext_keyans_show_item_opt:
4922              \skip_horizontal:n { -\l__enumext_fake_item_indent_viii_dim - \l__enumext_labelsep_vii
4923           }
4924           {
4925              \tl_use:N \l__enumext_fake_item_indent_viii_tl
4926           }
4927    }
```

(*End of definition for* __enumext_start_item_viii:w.)

__enumext_stop_item_viii: The __enumext_stop_item_viii: function will finish the fetching \item and its ⟨*content*⟩ by closing the minipage environment and the *horizontal box*. Here we will reduce the *warnings* a bit by setting the value of \hbadness to 10000, print the ⟨*contents*⟩ of the *box* along with \footnote and finally set the vertical and horizontal spaces between rows and columns.

```
4928 \cs_new_protected_nopar:Nn \__enumext_stop_item_viii:
4929    {
4930       \__enumext_endminipage:
4931       \__enumext_stop_list_tag:n {keyans*}
4932       \hbox_set_end:
4933       \int_set:Nn \hbadness { 10000 }
4934       \box_use_drop:N \l__enumext_item_text_viii_box
4935       \IfDocumentMetadataTF { }
4936         {
4937            \bool_if:NF \l__enumext_footnotes_key_bool
4938              {
4939                 \__enumext_print_footnote:
4940              }
4941         }
```

```
4942      \int_compare:nNnTF
4943        { \l__enumext_item_column_pos_viii_int } = { \l__enumext_columns_viii_int }
4944        {
4945          \par\noindent
4946          \int_zero:N \l__enumext_item_column_pos_viii_int
4947        }
4948        {
4949          \skip_horizontal:N \l__enumext_columns_sep_viii_dim
4950        }
4951    }
```

(*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.

```
4952  \cs_new_protected:Nn \__enumext_remove_extra_parsep_viii:
4953    {
4954      \int_compare:nNnT
4955        {
4956          \int_mod:nn
4957            { \g__enumext_item_count_all_viii_int }
4958            { \l__enumext_columns_viii_int }
4959        }
4960        =
4961        { 0 }
4962        {
4963          \para_end:
4964          \skip_vertical:n { -\l__enumext_itemsep_viii_skip }
4965          \skip_vertical:N \c_zero_skip
4966          \int_gzero:N \g__enumext_item_count_all_viii_int
4967        }
4968    }
```

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

12.45   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.

```
4969  \NewDocumentCommand \getkeyans { m }
4970    {
4971      \exp_args:Ne \__enumext_getkeyans_aux:n
4972        { \tl_to_str:e { \text_expand:n {#1} } }
4973    }
```

(*End of definition for* `\getkeyans`*. This function is documented on page* 16*.*)

`\__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.

```
4974  \cs_new_protected:Npn \__enumext_getkeyans_aux:n #1
4975    {
4976      \str_if_in:nnTF {#1} { : }
4977        {
4978          \use:e
4979            {
4980              \cs_set:Npn \exp_not:N \__enumext_tmp:w ##1 \c_colon_str ##2 \scan_stop:
4981                { {##1} {##2} }
4982            }
4983          \exp_after:wN \__enumext_getkeyans:nn \__enumext_tmp:w #1 \scan_stop:
4984        }
4985        { \msg_error:nnn { enumext } { missing-colon } {#1} }
4986    }
```

(*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*⟩.

```
4987  \cs_new_protected:Npn \__enumext_getkeyans:nn #1 #2
4988    {
4989      \prop_if_exist:cTF { g__enumext_#1_prop }
```

```
4990        {
4991          \prop_item:cn { g__enumext_#1_prop }{#2}
4992        }
4993        {
4994          \msg_error:nnn { enumext } { undefined-storage-anskey } {#1}
4995        }
4996      }
```

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

12.46 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*⟩].

```
4997 \keys_define:nn  { enumext / print }
4998   {
4999     print*  .code:n     = \keys_precompile:neN { enumext / enumext* }
5000                            { \__enumext_filter_save_key:n {#1} }
5001                            \l__enumext_print_keyans_starred_tl, % starred cmd
5002     print*  .initial:n = { nosep, label=\arabic*., columns=2, first=\small, font=\small },
5003     print-1 .code:n     = \keys_precompile:neN { enumext / level-1 }
5004                            { \__enumext_filter_save_key:n {#1} }
5005                            \l__enumext_print_keyans_i_tl,
5006     print-1 .initial:n = { nosep, label=\arabic*., columns=2, first=\small, font=\small },
5007     print-2 .code:n     = \keys_precompile:neN { enumext / level-2 }
5008                            { \__enumext_filter_save_key:n {#1} }
5009                            \l__enumext_print_keyans_ii_tl,
5010     print-2 .initial:n = { nosep, label=(\alph*), first=\small, font=\small },
5011     print-3 .code:n     = \keys_precompile:neN { enumext / level-3 }
5012                            { \__enumext_filter_save_key:n {#1} }
5013                            \l__enumext_print_keyans_iii_tl,
5014     print-3 .initial:n = { nosep, label=\roman*., first=\small, font=\small },
5015     print-4 .code:n     = \keys_precompile:neN { enumext / level-4 }
5016                            { \__enumext_filter_save_key:n {#1} }
5017                            \l__enumext_print_keyans_iv_tl,
5018     print-4 .initial:n = { nosep, label=\Alph*., first=\small, font=\small },
5019     print-* .code:n     = \keys_precompile:neN { enumext / enumext* }
5020                            { \__enumext_filter_save_key:n {#1} }
5021                            \l__enumext_print_keyans_vii_tl, % starred nested
5022     print-* .initial:n = { nosep, label=\arabic*., first=\small, font=\small },
5023   }
```

🎯 The reason for storing ⟨*keys*⟩ in token lists using \keys_precompile:neN is because the keys are set via \setenumext but are later executed by running the command \printkeyans and they are not handled directly by its optional argument, except those related to the first opening level.

\printkeyans   Create a user command to print *"all stored content"* in ⟨*sequence*⟩ for \anskey, anskey*, \item* and \anspic*. Within a group we will run our *"precompiled keys"* and then call the internal function __enumext_printkeyans:nnn.

```
5024 \NewDocumentCommand \printkeyans { s O{} m }
5025   {
5026     \group_begin:
5027       \tl_use:N \l__enumext_print_keyans_i_tl
5028       \tl_use:N \l__enumext_print_keyans_ii_tl
5029       \tl_use:N \l__enumext_print_keyans_iii_tl
5030       \tl_use:N \l__enumext_print_keyans_iv_tl
5031       \tl_use:N \l__enumext_print_keyans_vii_tl
5032       \__enumext_printkeyans:nnn { #1 } { #2 } { #3 }
5033     \group_end:
5034   }
```

(*End of definition for* \printkeyans. *This function is documented on page 17.*)

__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.

```
5035 \cs_new_protected:Npn \__enumext_printkeyans:nnn #1 #2 #3
5036   {
5037     \seq_if_exist:cTF { g__enumext_#3_seq }
5038       {
5039         \seq_if_empty:cF { g__enumext_#3_seq }
5040           {
5041             %%\seq_show:c { g__enumext_#3_seq }
```

If the starred argument is present we will check that the environment enumext* is not saved in the ⟨*sequence*⟩, then execute the variable \\l__enumext_print_keyans_starred_tl that contains the default ⟨*keys*⟩ for the environment enumext*, it will open the environment enumext* passing the optional argument to the *"first level"*, set the key base-fix and then will map the ⟨*sequence*⟩.

```
5042               \bool_if:nTF {#1}
5043                 {
5044                   \seq_if_in:cnTF { g__enumext_#3_seq } { \end{enumext*} }
5045                     {
5046                       \msg_error:nnnn { enumext } { print-starred } {#3} { enumext* }
5047                     }
5048                     {
5049                       \tl_use:N \l__enumext_print_keyans_starred_tl
5050                       % \IfDocumentMetadataTF { \tag_tool:n {para/flattened} } { }
5051                       \begin{enumext*}[#2]
5052                         \keys_set:nn { enumext / level-1 }{ base-fix }
5053                         \seq_map_inline:cn { g__enumext_#3_seq } { ##1 }
5054                       \end{enumext*}
5055                     }
5056                 }
```

Otherwise it will open the environment enumext passing the optional argument to the *"first level"*, set the key base-fix and then map the ⟨*sequence*⟩.

```
5057                 {
5058                   \begin{enumext}[#2]
5059                     \keys_set:nn { enumext / enumext* }{ base-fix }
5060                     \seq_map_inline:cn { g__enumext_#3_seq } { ##1 }
5061                   \end{enumext}
5062                 }
5063             }
5064       }
5065       {
5066         \msg_error:nnn { enumext } { undefined-storage-anskey } {#3}
5067       }
5068   }
```

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

12.47   The command \setenumext

The command \setenumext will be in charge of managing the ⟨*keys*⟩ passed to all environments and to the \printkeyans command. We must take precautions with the enumext* environment and *"first level"* of the enumext environment so as not to capture ⟨*keys*⟩ that complicate us.

__enumext_filter_first_level:n
__enumext_filter_first_level_key:n
__enumext_filter_first_level_pair:nn

The function __enumext_filter_first_level:n will be in charge of filtering the ⟨*keys*⟩ passed to the environment enumext* and *"first level"* of the environment enumext.

```
5069 \cs_new:Npn \__enumext_filter_first_level:n #1
5070   {
5071     \use:e
5072       {
5073         \keyval_parse:NNn
5074           \__enumext_filter_first_level_key:n
5075           \__enumext_filter_first_level_pair:nn {#1}
5076       }
5077   }
```

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

```
5078 \cs_new:Npn \__enumext_filter_first_level_key:n #1
5079   {
5080     \str_case:nnF {#1}
5081       {
5082         { resume    } {}
```

```
5083          { resume*  } {}
5084        }
5085        { , { \exp_not:n {#1} } }
5086    }
```

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

```
5087 \cs_new:Npn \__enumext_filter_first_level_pair:nn #1#2
5088    {
5089      \str_case:nnF {#1}
5090        {
5091          { series } {}
5092          { resume } {}
5093          { save-ans } {}
5094        }
5095        { , { \exp_not:n {#1} } = { \exp_not:n {#2} } }
5096    }
```

(*End of definition for* `\__enumext_filter_first_level:n`, `\__enumext_filter_first_level_key:n`, *and* `\__enumext_fil-ter_first_level_pair:nn`.)

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

```
5097 \keys_define:nn { enumext / meta-families }
5098    {
5099      enumext-1 .code:n =
5100                {
5101                  \keys_set:ne { enumext / level-1 }
5102                    {
5103                      \__enumext_filter_first_level:n {#1}
5104                    }
5105                } ,
5106      enumext-2 .code:n = { \keys_set:nn { enumext / level-2 } {#1} } ,
5107      enumext-3 .code:n = { \keys_set:nn { enumext / level-3 } {#1} } ,
5108      enumext-4 .code:n = { \keys_set:nn { enumext / level-4 } {#1} } ,
5109      keyans    .code:n = { \keys_set:nn { enumext / keyans  } {#1} } ,
5110      enumext*  .code:n =
5111                {
5112                  \keys_set:ne { enumext / enumext* }
5113                    {
5114                      \__enumext_filter_first_level:n {#1}
5115                    }
5116                } ,
5117      keyans*   .code:n = { \keys_set:nn { enumext / keyans* } {#1} } ,
5118      print*    .code:n = { \keys_set:nn { enumext / print   } { print*  = {#1} } } ,
5119      print-1   .code:n = { \keys_set:nn { enumext / print   } { print-1 = {#1} } } ,
5120      print-2   .code:n = { \keys_set:nn { enumext / print   } { print-2 = {#1} } } ,
5121      print-3   .code:n = { \keys_set:nn { enumext / print   } { print-3 = {#1} } } ,
5122      print-4   .code:n = { \keys_set:nn { enumext / print   } { print-4 = {#1} } } ,
5123      print-*   .code:n = { \keys_set:nn { enumext / print   } { print-* = {#1} } } ,
5124      unknown   .code:n = { \msg_error:nn { enumext } { unknown-key-family } } ,
5125    }
```

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

```
5126 \seq_const_from_clist:Nn \c__enumext_all_families_seq
5127    {
5128      enumext-1, enumext-2, enumext-3, enumext-4, keyans, enumext*,
5129      keyans*, print-1, print-2, print-3, print-4, print-*, print*,
5130    }
```

`\setenumext`   Now we define the user command `\setenumext`.

```
5131 \NewDocumentCommand \setenumext { O{enumext,1} +m }
5132    {
5133      \seq_clear:N \l__enumext_setkey_tmpa_seq
5134      \seq_set_from_clist:Nn \l__enumext_setkey_tmpb_seq {#1}
5135      \int_set:Nn \l__enumext_setkey_tmpa_int
5136        {
5137          \seq_count:N \l__enumext_setkey_tmpb_seq
5138        }
5139      \int_compare:nNnTF { \l__enumext_setkey_tmpa_int } > { 1 }
5140        {
5141          \seq_pop_left:NN \l__enumext_setkey_tmpb_seq \l__enumext_setkey_tmpa_tl
5142          \seq_map_function:NN \l__enumext_setkey_tmpb_seq \__enumext_set_parse:n
```

```
5143        \seq_set_map_e:NNn \l__enumext_setkey_tmpa_seq \l__enumext_setkey_tmpa_seq
5144          {
5145            \tl_use:N \l__enumext_setkey_tmpa_tl - ##1
5146          }
5147        }
5148        {
5149          \seq_put_right:Ne \l__enumext_setkey_tmpa_seq { \tl_trim_spaces:n {#1} }
5150        }
5151      \seq_if_empty:NTF \l__enumext_setkey_tmpa_seq
5152        { \seq_map_inline:Nn \c__enumext_all_families_seq }
5153        { \seq_map_inline:Nn \l__enumext_setkey_tmpa_seq }
5154        {
5155          \keys_set:nn { enumext / meta-families } { ##1 = {#2} }
5156        }
5157    }
```

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

__enumext_set_parse:n

__enumext_set_error:nn

Internal functions used by the \setenumext command.

```
5158 \cs_new_protected:Npn \__enumext_set_parse:n #1
5159    {
5160      \tl_set:Ne \l__enumext_setkey_tmpb_tl { \tl_trim_spaces:n {#1} }
5161      \clist_map_inline:nn { 0, 1, 2, 3, 4, * } % <- max level
5162        { \tl_remove_all:Nn \l__enumext_setkey_tmpb_tl {##1} }
5163      \tl_if_empty:NTF \l__enumext_setkey_tmpb_tl
5164        {
5165          \seq_put_right:Ne \l__enumext_setkey_tmpa_seq
5166            { \tl_trim_spaces:n {#1} }
5167        }
5168        { \__enumext_set_error:nn {#1} { } }
5169    }
5170 \cs_new_protected:Npn \__enumext_set_error:nn #1 #2
5171    { \msg_error:nnn { enumext } { invalid-key } {#1} {#2} }
```

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

12.48 The command \setenumextmeta

The command \setenumextmeta will be responsible for adding new *"meta-keys"* for the enumext and enumext* environments. The implementation code was given by Jonathan P. Spratte (@Skillmon) answer in Add .meta key to existing keys (l3keys).

\setenumextmeta

\c__enumext_meta_paths_prop

__enumext_add_meta_key:nnn

__enumext_def_meta_key:nnn

__enumext_def_meta_key:Vnn

First we will create a prop list \c__enumext_meta_paths_prop to handle the optional argument.

```
5172 \prop_const_from_keyval:Nn \c__enumext_meta_paths_prop
5173    {
5174      {enumext,1} = level-1,
5175      {enumext,2} = level-2,
5176      {enumext,3} = level-3,
5177      {enumext,4} = level-4,
5178      {enumext*}  = enumext*
5179    }
```

Now we create the user command taking care that unknown cannot be passed as an argument.

```
5180 \NewDocumentCommand \setenumextmeta { s O{enumext,1} m +m }
5181    {
5182      \str_if_eq:eeTF { \tl_trim_spaces:n {#3} } { unknown }
5183        { \msg_error:nn { enumext } { prohibited-unknown } }
5184        {
5185          \bool_if:nTF {#1}
5186            {
5187              \int_step_inline:nn { 4 }
5188                { \__enumext_add_meta_key:nnn { enumext, ##1 } {#3} {#4} }
5189              \__enumext_add_meta_key:nnn { enumext* } {#3} {#4}
5190            }
5191            { \__enumext_add_meta_key:nnn {#2} {#3} {#4} }
5192        }
5193    }
```

The internal functions __enumext_add_meta_key:nnn and __enumext_def_meta_key:nnn will check the optional argument and create the *"meta-key"*.

```
5194 \cs_new_protected:Npn \__enumext_add_meta_key:nnn #1
5195    {
5196      \tl_set:Nn \l__enumext_meta_path_tl {#1}
```

```
5197        \tl_replace_all:Nnn \l__enumext_meta_path_tl { ~ } {}
5198        \prop_get:NVNTF
5199          \c__enumext_meta_paths_prop \l__enumext_meta_path_tl \l__enumext_meta_path_tl
5200          { \__enumext_def_meta_key:Vnn \l__enumext_meta_path_tl }
5201          {
5202            \msg_error:nnn { enumext } { unknown-set } {#1}
5203            \use_none:nn
5204          }
5205      }
5206 \cs_new_protected:Npn \__enumext_def_meta_key:nnn #1#2#3
5207    {
5208      \bool_lazy_or:nnTF
5209        { \keys_if_exist_p:nn { enumext / #1 } {#2} }
5210        { \keys_if_exist_p:nn { enumext / enumext* } {#2} }
5211        { \msg_error:nnn { enumext } { already-defined } {#2} }
5212        {
5213          \keys_define:nn { enumext / #1 }
5214            {
5215              #2 .meta:n = {#3},
5216              #2 .value_forbidden:n = true
5217            }
5218        }
5219    }
5220 \cs_generate_variant:Nn \__enumext_def_meta_key:nnn { V }
```

(*End of definition for* \setenumextmeta *and others. This function is documented on page* 7.)

12.49 The command \foreachkeyans

The command \foreachkeyans will execute a *loop* over the ⟨*prop list*⟩ and return its contents. The implementation code is adapted from the answer provided by Enrico Gregorio (@egreg) in Expand a .cs defined by key inside the function.

\foreachkeyans
__enumext_parse_foreach_keys:nn
__enumext_parse_foreach_keys:n
__enumext_foreach_keyans:nn
__enumext_foreach_add_body:n

We define a set of ⟨*keys*⟩ for command and we will save the default values of these in \g__enumext_-foreach_default_keys_tl to avoid the use of group.

```
5221 \keys_define:nn { enumext / foreach }
5222    {
5223      before  .tl_set:N  = \l__enumext_foreach_before_tl,
5224      before  .value_required:n = true,
5225      after   .tl_set:N  = \l__enumext_foreach_after_tl,
5226      after   .value_required:n = true,
5227      start   .int_set:N = \l__enumext_foreach_start_int,
5228      start   .value_required:n = true,
5229      stop    .int_set:N = \l__enumext_foreach_stop_int,
5230      stop    .value_required:n = true,
5231      step    .int_set:N = \l__enumext_foreach_step_int,
5232      step    .value_required:n = true,
5233      wrapper .cs_set_protected:Np = \__enumext_foreach_wrapper:n #1,
5234      wrapper .value_required:n = true,
5235      sep     .tl_set:N  = \l__enumext_foreach_sep_tl,
5236      sep     .value_required:n = true,
5237      unknown .code:n    = { \__enumext_parse_foreach_keys:n {#1} }
5238    }
5239 \keys_precompile:nnN { enumext / foreach }
5240    {
5241      before={},after={},start=1,step=1,stop=0,wrapper#1,sep=
5242    }
5243    \g__enumext_foreach_default_keys_tl
```

Functions for handling unknown ⟨*keys*⟩.

```
5244 \cs_new_protected:Npn \__enumext_parse_foreach_keys:nn #1#2
5245    {
5246      \tl_if_blank:nTF {#2}
5247        {
5248          \msg_error:nnn { enumext } { for-key-unknown } {#1}
5249        }
5250        {
5251          \msg_error:nnnn { enumext } { for-key-value-unknown } {#1} {#2}
5252        }
5253    }
5254 \cs_new_protected:Npn \__enumext_parse_foreach_keys:n #1
5255    {
```

```
5256        \exp_args:NV \__enumext_parse_foreach_keys:nn \l_keys_key_str {#1}
5257      }
```

We create the command.

```
5258  \NewDocumentCommand \foreachkeyans { +O{} m }
5259    {
5260      \__enumext_foreach_keyans:nn {#1} {#2}
5261    }
```

Finally the internal functions `\__enumext_foreach_keyans:nn` and `\__enumext_foreach_add_body:n` will loop through the prop list and print the contents.

```
5262  \cs_new_protected:Npn \__enumext_foreach_keyans:nn #1 #2
5263    {
5264      \tl_use:N \g_enumext_foreach_default_keys_tl
5265      \keys_set:nn { enumext / foreach } {#1}
5266      \tl_set:Nn \l__enumext_foreach_name_prop_tl {#2}
5267      \prop_if_exist:cF { g__enumext_#2_prop }
5268        {
5269          \msg_error:nnn { enumext } { undefined-storage-anskey } {#2}
5270        }
5271      \int_compare:nNnT { \l__enumext_foreach_stop_int } = { 0 }
5272        {
5273          \int_set:Nn \l__enumext_foreach_stop_int
5274            { \prop_count:c { g__enumext_#2_prop } }
5275        }
5276      \seq_clear:N \l__enumext_foreach_print_seq
5277      \int_step_function:nnnN
5278        { \l__enumext_foreach_start_int }
5279        { \l__enumext_foreach_step_int }
5280        { \l__enumext_foreach_stop_int }
5281        \__enumext_foreach_add_body:n
5282        \seq_use:NV \l__enumext_foreach_print_seq \l__enumext_foreach_sep_tl
5283    }
5284  \cs_new_protected:Npn \__enumext_foreach_add_body:n #1
5285    {
5286      \seq_put_right:Ne \l__enumext_foreach_print_seq
5287        {
5288          \exp_not:V \l__enumext_foreach_before_tl
5289          \__enumext_foreach_wrapper:n
5290            {
5291              \prop_item:cn { g__enumext_ \l__enumext_foreach_name_prop_tl _prop }{#1}
5292            }
5293          \exp_not:V \l__enumext_foreach_after_tl
5294        }
5295    }
```

(*End of definition for* `\foreachkeyans` *and others. This function is documented on page* *16*.)

12.50   Messages

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

```
5296  \msg_new:nnn { enumext } { package-load }
5297    {
5298      The ~ '#1' ~ package ~ is ~ already ~ loaded.
5299    }
5300  \msg_new:nnn { enumext } { package-not-load }
5301    {
5302      The ~ '#1' ~ package ~ will ~ be ~ loaded ~ as ~ a ~ dependency.
5303    }
5304  \msg_new:nnn { enumext } { package-load-foot }
5305    {
5306      The ~ '#1' ~ package ~ is ~ loaded ~ with ~ the ~ option ~ '#2'.
5307    }
```

Message used in the creation of counters by enumext package.

```
5308  \msg_new:nnn { enumext } { counters }
5309    {
5310      The ~ counter ~ '#1' ~ is ~ already ~ defined ~ by ~ some ~ \\
5311      package ~ or ~ macro, ~ it ~ cannot ~ be ~ continued.
5312    }
```

Message used by align and mark-pos keys.

```
5313  \msg_new:nnn { enumext } { unknown-choice }
```

```
5314      {
5315        The ~ value ~ '#3' ~ for ~ '#1' ~ key ~ is ~ invalid ~ use ~ ('#2').
5316      }
```

Message used by reserved anskey* environment by enumext package.

```
5317  \msg_new:nnnn { enumext } { anskey-env-error }
5318      {
5319        The ~ '#1' ~ environment ~is ~ reserved ~ by ~\\
5320        'enumext' ~ package, ~ It~ is~ already~ defined.
5321      }
5322      {
5323        The ~ anskey* ~ environment ~ is ~ defined ~ internally ~
5324        for ~ the ~ 'save-ans' ~ key.\\
5325      }
```

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

```
5326  \msg_new:nnn { enumext } { store-prop }
5327      {
5328        * ~ Package ~ enumext: ~ Creating ~
5329          \c_backslash_str g__enumext_#1_prop ~ \msg_line_context:.
5330      }
5331  \msg_new:nnn { enumext } { store-seq }
5332      {
5333        * ~ Package ~ enumext: ~ Creating ~
5334        \c_backslash_str g__enumext_#1_seq ~ \msg_line_context:.
5335      }
5336  \msg_new:nnn { enumext } { store-int }
5337      {
5338        * ~ Package ~ enumext: ~ Creating ~
5339        \c_backslash_str g__enumext_resume_#1_int ~ \msg_line_context:.
5340      }
5341  \msg_new:nnn { enumext } { prop-seq-int-hook }
5342      {
5343        * ~ Package ~ enumext: ~ Elements ~ in ~
5344        \c_backslash_str g__enumext_#1_prop ~ = ~ #2.\\
5345        * ~ Package ~ enumext: ~ Elements ~ in ~
5346        \c_backslash_str g__enumext_#1_seq ~ = ~ #3.\\
5347        * ~ Package ~ enumext: ~ Value ~ off ~
5348        \c_backslash_str g__enumext_resume_#1_int ~ = ~ #4.
5349      }
5350  \msg_new:nnn { enumext } { item-answer-hook }
5351      {
5352        * ~ Package ~ enumext: ~ Value ~ off ~
5353        \c_backslash_str g__enumext_item_number_int ~ = ~ #1.\\
5354        * ~ Package ~ enumext: ~ Value ~ off ~
5355        \c_backslash_str g__enumext_item_anskey_int ~ = ~ #2.\\
5356        * ~ Package ~ enumext: ~ Difference ~ item_number_int ~ - ~ item_anskey_int ~ = ~ #3.
5357      }
```

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

```
5358  \msg_new:nnn { enumext } { invalid-key }
5359      {
5360        The ~ key ~ '#1' ~ is ~ not ~ know ~ the ~ level ~ #2.
5361      }
5362  \msg_new:nnn { enumext } { unknown-key-family }
5363      {
5364        Unknown~key~family~`\l_keys_key_str'~for~enumext.
5365      }
```

Messages used in length calculation.

```
5366  \msg_new:nnn { enumext } { width-negative }
5367      {
5368        Ignoring ~ negative ~ value ~ '#1=#2' ~ \msg_line_context:.\\
5369        The ~ key ~ '#1'~ accepts ~ values  ~ >= ~ 0pt.
5370      }
5371  \msg_new:nnn { enumext } { width-zero }
5372      {
5373        Invalid ~ '#1=#2' ~ \msg_line_context:.\\
5374        The ~ key ~ '#1'~ accepts ~ values  ~ > ~ 0pt.
5375      }
```

Messages used by show-length key in enumext.

```
5376  \msg_new:nnn { enumext } { list-lengths }
```

```
5377    {
5378      **** ~ Lengths ~ used ~ by ~ 'enumext' ~ level ~ '#2' ~ \msg_line_context:~\c_space_tl ****\\
5379      \__enumext_show_length:nnn { dim  } { labelsep     } {#1}
5380      \__enumext_show_length:nnn { dim  } { labelwidth   } {#1}
5381      \__enumext_show_length:nnn { dim  } { itemindent   } {#1}
5382      \__enumext_show_length:nnn { dim  } { leftmargin   } {#1}
5383      \__enumext_show_length:nnn { dim  } { rightmargin  } {#1}
5384      \__enumext_show_length:nnn { dim  } { listparindent } {#1}
5385      \__enumext_show_length:nnn { skip } { topsep    } {#1}
5386      \__enumext_show_length:nnn { skip } { parsep    } {#1}
5387      \__enumext_show_length:nnn { skip } { partopsep } {#1}
5388      \__enumext_show_length:nnn { skip } { itemsep   } {#1}
5389      ****************************************************
5390    }
```

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

```
5391  \msg_new:nnn { enumext } { list-lengths-not-nested }
5392    {
5393      **** ~ Lengths ~ used ~ by ~ '#2' ~ environment ~ \msg_line_context:~\c_space_tl ****\\
5394      \__enumext_show_length:nnn { dim  } { labelsep     } {#1}
5395      \__enumext_show_length:nnn { dim  } { labelwidth   } {#1}
5396      \__enumext_show_length:nnn { dim  } { itemindent   } {#1}
5397      \__enumext_show_length:nnn { dim  } { leftmargin   } {#1}
5398      \__enumext_show_length:nnn { dim  } { rightmargin  } {#1}
5399      \__enumext_show_length:nnn { dim  } { listparindent } {#1}
5400      \__enumext_show_length:nnn { skip } { topsep    } {#1}
5401      \__enumext_show_length:nnn { skip } { parsep    } {#1}
5402      \__enumext_show_length:nnn { skip } { partopsep } {#1}
5403      \__enumext_show_length:nnn { skip } { itemsep   } {#1}
5404      ****************************************************
5405    }
```

Messages used by ref key.

```
5406  \msg_new:nnn { enumext } { key-ref-empty }
5407    {
5408      Key ~ 'ref' ~ need ~ a ~ value ~ in ~ '#1'~ \msg_line_context:.
5409    }
```

Messages used by save-ans key.

```
5410  \msg_new:nnn { enumext } { save-ans-empty }
5411    {
5412      Key ~ 'save-ans' ~ need ~ a ~ value ~ in ~ '#1'~ \msg_line_context:.
5413    }
5414  \msg_new:nnn { enumext } { save-ans-log }
5415    {
5416      * ~ Package ~ enumext: ~ Start ~ #1\c_space_tl with ~ save-ans=#2 ~ \msg_line_context:.
5417    }
5418  \msg_new:nnn { enumext } { save-ans-log-hook }
5419    {
5420      * ~ Package ~ enumext: ~ Stop ~ #1\c_space_tl with ~ save-ans=#2 ~ \msg_line_context:.
5421    }
5422  \msg_new:nnn { enumext } { save-ans-hook }
5423    {
5424      Stop ~ storing ~ for ~ 'save-ans=#1' ~ \msg_line_context:.
5425    }
```

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

```
5426  \msg_new:nnn { enumext } { need-save-ans }
5427    {
5428      Key ~ '#1'~ works ~ only ~ with ~ the ~ 'save-ans' ~ key ~ in ~ '#2'~ \msg_line_context:.
5429    }
5430  \msg_new:nnn { enumext } { items-same-answer }
5431    {
5432      ****************************************\\
5433      * ~ Package ~ enumext: ~ Checking ~ answers ~ in ~ '#1' ~
5434      for ~ \c_left_brace_str #2 \c_right_brace_str\\
5435      * ~ started ~ #3 ~ and ~ close ~ \msg_line_context: : ~
5436      'OK', ~ all ~ items ~ with ~ answer.\\
5437      ****************************************
5438    }
5439  \msg_new:nnn { enumext } { item-greater-answer }
5440    {
5441      Checking ~ answers ~ in ~ '#1' ~ for ~ \c_left_brace_str #2 \c_right_brace_str\\
```

```
5442      started ~ #3 ~ and ~ close ~ \msg_line_context: : ~'NOT ~ OK'\\
5443      Items ~ > ~ Answers.
5444    }
5445  \msg_new:nnn { enumext } { item-less-answer }
5446    {
5447      Checking ~ answers ~ in ~ '#1' ~ for ~ \c_left_brace_str #2 \c_right_brace_str\\
5448      started ~ #3 ~ and ~ close ~ \msg_line_context: : ~'NOT ~ OK'\\
5449      Items ~ < ~ Answers.
5450    }
```

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

```
5451  \msg_new:nnn { enumext } { missing-starred }
5452    {
5453      Missing ~ '\c_backslash_str #1*' ~ #2.
5454    }
5455  \msg_new:nnn { enumext } { many-starred }
5456    {
5457      Many ~ '\c_backslash_str #1*' ~ #2.
5458    }
```

Messages used by \printkeyans* command.

```
5459  \msg_new:nnn { enumext } { print-starred }
5460    {
5461      \c_backslash_str printkeyans*:~ The ~ sequence ~ '#1' ~ already ~ contains ~
5462      #2 ~ environment ~  \msg_line_context:.
5463    }
```

Message for the nesting depth of the environment enumext.

```
5464  \msg_new:nnn { enumext } { list-too-deep }
5465    {
5466      Too ~ deep ~ nesting  ~ for ~ 'enumext' ~ \msg_line_context:.~ \\
5467      The ~ maximum  ~ level  ~ of  ~ nesting  ~ is ~ 4.
5468    }
```

Messages used by \anskey, anskey* and \anspic commands.

```
5469  \msg_new:nnn { enumext } { anskey-unnumber-item }
5470    {
5471      Can't ~ store ~ with ~ a ~ unnumbered ~ \c_backslash_str item ~ \msg_line_context:.
5472    }
5473  \msg_new:nnn { enumext } { anskey-already-stored }
5474    {
5475      Content ~ already ~ stored ~ for ~ this ~ \c_backslash_str item ~ \msg_line_context:.
5476    }
5477  \msg_new:nnn { enumext } { anskey-empty-arg }
5478    {
5479      Can't ~ store ~ empty ~ content ~ \msg_line_context:.
5480    }
5481  \msg_new:nnn { enumext } { anskey-wrong-place }
5482    {
5483      Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:.~ \\
5484      '\c_backslash_str #1' ~ works ~ in ~ the ~ environment ~ '#2'.
5485    }
5486  \msg_new:nnn { enumext } { anskey-nested }
5487    {
5488      The ~ command ~ \c_backslash_str anskey~ can't ~ be ~ nested ~ \msg_line_context:.
5489    }
5490  \msg_new:nnn { enumext } { anskey-math-mode }
5491    {
5492      #1 ~ can't ~ work ~ in ~ math ~ mode ~ \msg_line_context:.
5493    }
5494  \msg_new:nnn { enumext } { anskey-env-wrong }
5495    {
5496      The ~ environment ~ anskey* ~ cannot ~ use ~ in ~ '#1' ~ \msg_line_context:.
5497    }
5498  \msg_new:nnn { enumext } { anspic-wrong-place }
5499    {
5500      Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:.~ \\
5501      '\c_backslash_str #1' ~ works ~ in ~ the ~ environment ~ '#2'.
5502    }
5503  \msg_new:nnn { enumext } { command-wrong-place }
5504    {
5505      Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:.~ \\
5506      '\c_backslash_str #1' ~ works ~ outside ~ the ~ environment ~ '#2'.
```

```
5507      }
5508  \msg_new:nnnn { enumext } { anskey-env-key-unknown }
5509      {
5510        The ~ key ~ '#1' ~ is ~ unknown ~ by ~ environment~
5511        'anskey*' ~ and ~ is ~ being ~ ignored.
5512      }
5513      {
5514        The ~ environment ~ 'anskey*' ~ does ~ not ~ have ~ a ~ key ~ called ~'#1'.\\
5515        Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5516      }
5517  \msg_new:nnnn { enumext } { anskey-env-key-value-unknown }
5518      {
5519        The ~ key ~ '#1=#2' ~ is ~ unknown ~ by ~ environment ~
5520        'anskey*' ~ and ~ is ~ being ~ ignored.
5521      }
5522      {
5523        The ~ environment ~ 'anskey*' ~ does ~ not ~ have ~ a ~ key ~ called ~'#1'.\\
5524        Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5525      }
5526  \msg_new:nnnn { enumext } { anskey-cmd-key-unknown }
5527      { The ~ key ~'#1'~ is ~ unknown ~ by ~ '\c_backslash_str anskey' ~ and ~ is ~ being ~ ignored.}
5528      {
5529        The ~ command ~'\c_backslash_str anskey' ~ does ~ not ~ have ~ a ~ key ~ called ~'#1'.\\
5530        Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5531      }
5532  \msg_new:nnnn { enumext } { anskey-cmd-key-value-unknown }
5533      { The ~ key ~ '#1=#2' ~ is ~ unknown ~ by ~ '\c_backslash_str anskey' ~ and ~ is ~ being ~ ignor
5534      {
5535        The ~ command ~ '\c_backslash_str anskey' ~ does ~ not ~ have ~ a ~ key ~ called ~'#1'.\\
5536        Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5537      }
```

Messages used by keyans, keyans* and keyanspic environment.

```
5538  \msg_new:nnn { enumext } { keyans-nested }
5539      {
5540        The ~ environment ~ 'keyans' ~ can't ~ be  ~ nested  ~ \msg_line_context:.
5541      }
5542  \msg_new:nnn { enumext } { keyans-wrong-level }
5543      {
5544        Wrong ~ level ~ position ~ for ~ 'keyans' ~ \msg_line_context:.~ \\
5545        The ~ environment ~ 'keyans' ~ can ~ only ~ be ~ in ~ the ~ first ~ level.
5546      }
5547  \msg_new:nnn { enumext } { wrong-place }
5548      {
5549        Wrong ~ place ~ for ~ '#1' ~ environment ~\msg_line_context:.~ \\
5550        '#1' ~ is ~ only ~ found ~ with ~ '#2' ~  in  ~  'enumext.
5551      }
5552  \msg_new:nnn { enumext } { keyanspic-nested }
5553      {
5554        The ~ environment ~ 'keyanspic' ~ can't ~ be  ~ nested~ \msg_line_context:.~.
5555      }
5556  \msg_new:nnn { enumext } { keyanspic-wrong-level }
5557      {
5558        Wrong ~ level ~ position ~ for ~ 'keyanspic' ~ \msg_line_context:.~ \\
5559        The ~ environment ~ 'keyans' ~ can ~ only ~ be ~ in ~ the ~ first ~ level.
5560      }
5561  \msg_new:nnn { enumext } { keyanspic-item-cmd }
5562      {
5563        Can't ~ use  ~ \c_backslash_str item ~ in ~ keyanspic ~ \msg_line_context:.
5564      }
5565  \msg_new:nnnn { enumext } { keyans-unknown-key }
5566      {
5567        The ~ key ~ '#1' ~ is ~ unknown ~ by ~ environment~
5568        '\l__enumext_envir_name_tl' ~ and ~ is ~ being ~ ignored.
5569      }
5570      {
5571        The ~ environment ~ '\l__enumext_envir_name_tl' ~ does ~ not
5572        ~ have ~ a ~ key ~ called ~'#1'.\\
5573        Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5574      }
5575  \msg_new:nnnn { enumext } { keyans-unknown-key-value }
5576      {
```

```
5577       The ~ key ~ '#1=#2' ~ is ~ unknown ~ by ~ environment ~
5578       '\l__enumext_envir_name_tl' ~ and ~ is ~ being ~ ignored.
5579     }
5580     {
5581       The ~ environment ~ '\l__enumext_envir_name_tl' ~ does ~ not
5582       ~ have ~ a ~ key ~ called ~'#1'.\\
5583       Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5584     }
```

Message used by unknown ⟨*keys*⟩ in enumext*. environment.

```
5585  \msg_new:nnnn { enumext } { starred-unknown-key }
5586     {
5587       The ~ key ~ '#1' ~ is ~ unknown ~ by ~ environment~
5588       '\l__enumext_envir_name_tl' ~ and ~ is ~ being ~ ignored.
5589     }
5590     {
5591       The ~ environment ~ '\l__enumext_envir_name_tl' ~ does ~ not
5592       ~ have ~ a ~ key ~ called ~'#1'.\\
5593       Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5594     }
5595  \msg_new:nnnn { enumext } { starred-unknown-key-value }
5596     {
5597       The ~ key ~ '#1=#2' ~ is ~ unknown ~ by ~ environment ~
5598       '\l__enumext_envir_name_tl' ~ and ~ is ~ being ~ ignored.
5599     }
5600     {
5601       The ~ environment ~ '\l__enumext_envir_name_tl' ~ does ~ not
5602       ~ have ~ a ~ key ~ called ~'#1'.\\
5603       Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5604     }
```

Message used by unknown ⟨*keys*⟩ in enumext environment.

```
5605  \msg_new:nnnn { enumext } { standar-unknown-key }
5606     {
5607       The ~ key ~ '#1' ~ is ~ unknown ~ by ~ environment ~ '\l__enumext_envir_name_tl' \c_space_tl
5608       ~ on ~ level ~ \int_use:N \l__enumext_level_int \c_space_tl and ~ is ~ being ~ ignored.
5609     }
5610     {
5611       The ~ environment ~ '\l__enumext_envir_name_tl' ~ does ~ not
5612       ~ have ~ a ~ key ~ called ~'#1' ~ on ~ level ~ \int_use:N \l__enumext_level_int.\\
5613       Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5614     }
5615  \msg_new:nnnn { enumext } { standar-unknown-key-value }
5616     {
5617       The ~ key ~ '#1=#2' ~ is ~ unknown ~ by ~ environment ~ '\l__enumext_envir_name_tl' \c_space_t
5618       ~ on ~ level ~ \int_use:N \l__enumext_level_int \c_space_tl and ~ is ~ being ~ ignored.
5619     }
5620     {
5621       The ~ environment ~ '\l__enumext_envir_name_tl' ~ does ~ not
5622       ~ have ~ a ~ key ~ called ~'#1' ~ on ~ level ~ \int_use:N \l__enumext_level_int.\\
5623       Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5624     }
```

Message used by unknown ⟨*keys*⟩ in \foreachkeyans.

```
5625  \msg_new:nnnn { enumext } { for-key-unknown }
5626     { The~key~'#1'~is~unknown~by~'\c_backslash_str foreachkeyans'~and~is~being~ignored.}
5627     {
5628       The~command~'\c_backslash_str foreachkeyans'~does~not~have~a~key~called~'#1'.\\
5629       Check~that~you~have~spelled~the~key~name~correctly.
5630     }
5631  \msg_new:nnnn { enumext } { for-key-value-unknown }
5632     { The~key~'#1=#2'~is~unknown~by~'\c_backslash_str foreachkeyans'~and~is~being~ignored. }
5633     {
5634       The~command~'\c_backslash_str foreachkeyans'~does~not~have~a~key~called~'#1'.\\
5635       Check~that~you~have~spelled~the~key~name~correctly.
5636     }
```

Messages used by \getkeyans command.

```
5637  \msg_new:nnn { enumext } { undefined-storage-anskey }
5638     {
5639       Storage ~ named ~ '#1' ~ is ~ not ~ defined ~ \msg_line_context:.
5640     }
```

Messages used by `\miniright` command.

```
5641 \msg_new:nnn { enumext } { missing-miniright }
5642   {
5643     Missing ~ '\c_backslash_str miniright' ~ in ~ \msg_line_context:.\\
5644     The ~ key ~ 'mini-env' ~ need ~ '\c_backslash_str miniright'.
5645   }
5646 \msg_new:nnn { enumext } { wrong-miniright-place }
5647   {
5648     Wrong ~ place ~ for ~ '\c_backslash_str miniright' ~ \msg_line_context:.~ \\
5649     Works ~ in ~ 'enumext' ~ and ~ 'keyans' ~ with ~ key ~ 'mini-env'.
5650   }
5651 \msg_new:nnn { enumext } { wrong-miniright-use }
5652   {
5653     Wrong ~ use ~ for ~ '\c_backslash_str miniright' ~ \msg_line_context:.~ \\
5654     '\c_backslash_str miniright' ~ need ~ a ~ key ~ 'mini-env'.
5655   }
5656 \msg_new:nnn { enumext } { wrong-miniright-starred }
5657   {
5658     Can't ~ use  ~ \c_backslash_str miniright ~ in ~ starred ~ environments ~ \msg_line_context:.
5659   }
5660 \msg_new:nnn { enumext } { many-miniright-used }
5661   {
5662     Can't ~ use  ~ \c_backslash_str miniright ~ more ~ than ~ once ~  \msg_line_context:.
5663   }
```

Messages used by `\setenumextmeta` command.

```
5664 \msg_new:nnn { enumext } { unknown-set }
5665   {
5666     Argument ~ [#1] ~ is ~ unknown ~ by ~  \c_backslash_str setenumextmeta ~ \msg_line_context:.
5667   }
5668 \msg_new:nnn { enumext } { already-defined }
5669   {
5670     The ~ key ~ '#1' ~ is ~ already ~ defined ~ \msg_line_context:.
5671   }
5672 \msg_new:nnn { enumext } { prohibited-unknown }
5673   {
5674     The ~ name ~ 'unknown' ~ can't ~ be ~ chosen~ for ~ a ~ meta ~ key ~ \msg_line_context:.
5675   }
```

Messages used by `enumext*` and `keyans*` environments.

```
5676 \msg_new:nnn { enumext } { nested }
5677   {
5678     The ~ environment ~ \l__enumext_envir_name_tl \c_space_tl can't ~ be ~ nested ~ \msg_line_con
5679   }
5680 \msg_new:nnn { enumext } { nested-horizontal }
5681   {
5682     The ~ environment ~ \l__enumext_envir_name_tl \c_space_tl can't ~ be ~ nested ~ in ~ '#1' ~ '
5683   }
5684 \msg_new:nnn { enumext } { item-joined }
5685   {
5686     Items ~ joined ~ (#1) ~ > ~ #2  ~ columns ~\msg_line_context:.
5687   }
5688 \msg_new:nnn { enumext } { item-joined-columns }
5689   {
5690     Not ~ space ~ to ~ join ~ items ~ (#1) ~ > ~ #2 ~\msg_line_context:.
5691   }
```

12.51 Finish package

Finish package implementation.

```
5692 \file_input_stop:
5693 ⟨/package⟩
```

13 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.

V

`\value` 1781, 1787, 1794, 1800, 1808, 1814, 1821, 1827

vbox commands:

    `\vbox_set:Nn` 4061

    `\vbox_set_top:Nn` 4339, 4400

`\vspace` . 864, 875, 1617, 1620, 1631, 1634, 1644, 1646, 1655, 1657, 1666, 1668, 1677, 1679, 1688, 1690, 1699, 1701

W

`widest` 777

`wrap-ans` 2167

`wrap-label` 512

`wrap-label*` 512

`wrap-opt` 2167

Z

`\z` 2804