enumext

ENUMERATE EXERCISE SHEETS

v1.0 2024-06-13[*]

©2024 by Pablo González[†]

CTAN: https://www.ctan.org/pkg/enumext
https://github.com/pablgonz/enumext

Abstract

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

Contents

Motivation and acknowledgments

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

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

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

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

[*]This file describes a documentation for v1.0, last revised 2024-06-13.
[†]E-mail: «pablgonz@educarchile.cl».

License and Requirements

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

1 Introduction

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

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

1. Factor $x^2 - 2x + 1$
2. Factor $3x + 3y + 3z$
3. True False
   (a) $\alpha > \delta$
   (b) LaTeX2e is cool?
4. Related to Linux
   (a) You use linux?
   (b) Usually uses the package manager?
   (c) Rate the following package and class
      i. `xsim-exam`
      ii. `xsim`
      iii. `exsheets`

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

1. Factor $x^2 - 2x + 1$
   * $\boxed{(x-1)^2}$
2. Factor $3x + 3y + 3z$
   * $\boxed{3(x+y+z)}$
3. True False
   (a) $\alpha > \delta$
      * $\boxed{\text{False}}$
   (b) LaTeX2e is cool?
      * $\boxed{\text{Very True!}}$
4. Related to Linux
   (a) You use linux?
      * $\boxed{\text{Yes}}$
   (b) Usually uses the package manager?
      * $\boxed{\text{Yes, dnf}}$
   (c) Rate the following package and class
      i. `xsim-exam`
         * $\boxed{\text{doesn't exist for now :(}}$
      ii. `xsim`
         * $\boxed{\text{very good}}$
      iii. `exsheets`
         * $\boxed{\text{obsolete}}$

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

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

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

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

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

1. First type of questions

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

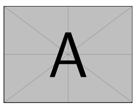
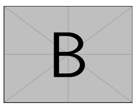
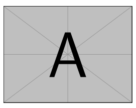
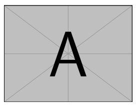
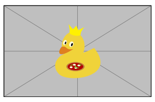
2. Second type of questions
   I. $2\alpha + 2\delta = 90^\circ$
   II. $\alpha = \delta$
   III. $\angle EDF = 45^\circ$

   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) $2\alpha + 2\delta = 90^\circ$
   (2) $\angle EDF = 45^\circ$

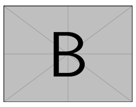
   A) value   D) value
   B) value   E) value
   C) value

4. Question with image and label below:


A)

B)

C)

D)

E)

5. Question with image on left side:

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



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

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

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

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

1.1 Description and usage

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

This package is NOT intend to replace the `enumerate` environment nor replace the powerful `enumitem`[6], the approach is intended to work without hindering either of them.
This package can be used with `xelatex`, `lualatex`, `pdflatex` and the classical `latex»dvips»ps2pdf` and is present in TeX Live and MiKTeX, use the package manager to install. For manual installation, download enumext.zip and unzip it, run `lualatex enumext.dtx` and move all files to appropriate locations, then run `mktexlsr`. To produce the documentation run `lualatex enumext.dtx` two times.

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

The package is loaded in the usual way:

```
\usepackage{enumext}
```

1.2 The concept of left margin

There is a direct relationship between the parameters `\leftmargin`, `\itemindent`, `\labelwidth` and `\labelsep` plus an *"extra space"* that makes it difficult to obtain the desired *horizontal spaces* in a `list` environment.
Usually we don't want the `list` to go beyond the left margin of the page, but since these four values are related, that causes a problem. The `enumitem`[6] package adds the `\labelindent` parameter to solve some of these problems. A simplified representation of this in the figure 1.

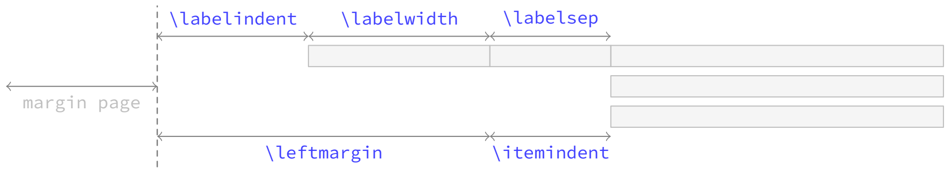


Figure 1: Representation of horizontal lengths in `enumitem`.

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

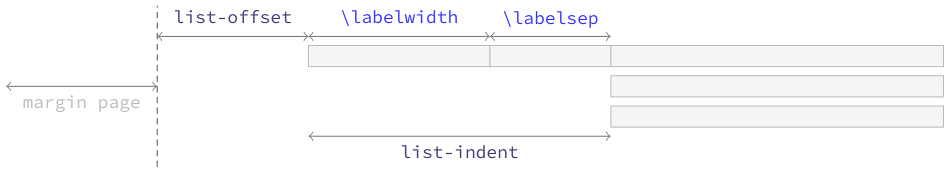


Figure 2: Representation of horizontal lengths concept in enumext.

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

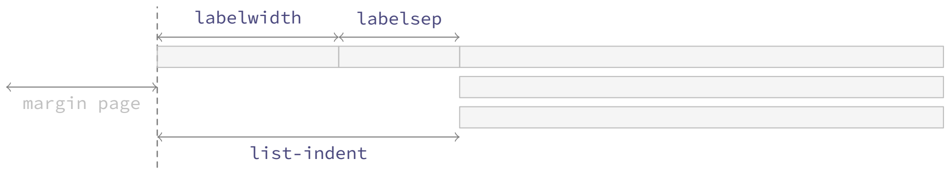


Figure 3: Default horizontal lengths `list-offset=0pt`, `list-indent=\labelwidth+\labelsep` in enumext.

1.3 User interface

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

1.3.1 Internal counters

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

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

1.3.2 Support for multicol

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

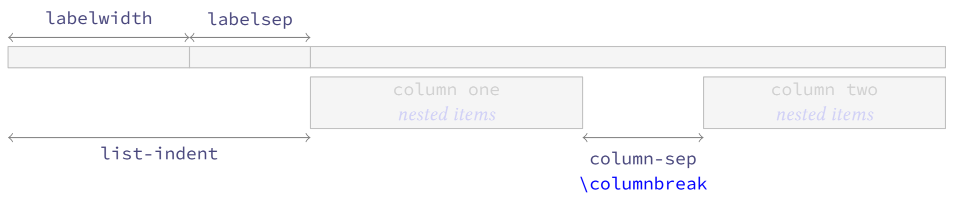


Figure 4: Representation of the two column output for a nested level in enumext environment.

The *"non starred"* version of the multicols environment is always used together with the \raggedcolumns command and is controlled by columns and columns-sep keys. It can be used in all nesting levels of the environment enumext and the environment keyans and can together with the mini-env key. If you need to force a start a new column \columnbreak must be used (see §4.5).

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

1.3.3 Support for minipage

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

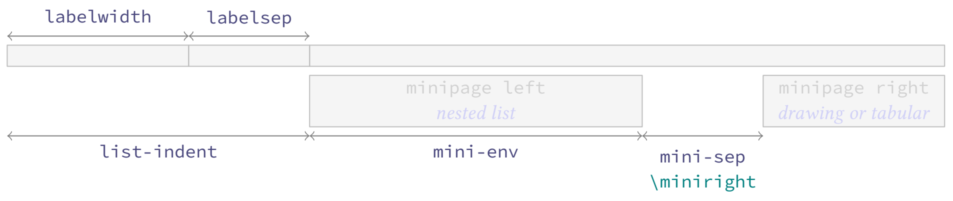


Figure 5: Representation of the mini-env output for a nested level enumext environment.

The minipage environments on *"left side"* and *"right side"* is always used with *"aligned on top"* [t]. It can be used in all nesting levels of the environment enumext and the environment keyans and is controlled by mini-env and mini-sep keys. In order to switch from the *"left"* side minipage environment to the *"right"* side one must use the command \miniright (see §4.6).

1.3.4 The \label and \ref system

This package provides a user interface like the enumitem[6] package to customize the references which is activated by the ref key (§4.1), the standard LaTeX \label and \ref commands work as usual. It also provides an *"internal reference"* system for the *"stored content"* by means of the key save-ref (§5.1.1) when the key save-ans (§5.1) is active.

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

1.3.5 Support for \footnote

This package provides an internal implementation for the \footnote command which is compatible with the hyperref package for the enumext* and keyans* environments, but will not produce the expected links, and if the mini-env key is used in enumext or keyans environments the output will look like the classic way they are displayed in the environment minipage.
The best way to solve this is to use Jean-François Burnol footnotehyper[9] package, it will support keeping the links if hyperref is loaded with the hyperfootnotes=true option (default) and will show the output numbered at the bottom of the page (as opposed to how it is displayed in the minipage environment). The way to load it is as follows:

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

2 The environments provided

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

enumext
enumext*

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

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

2.1 The environment enumext

The enumext is an environment that works in the same way as the standard enumerate environment provided by LaTeX, \item and \item[⟨custom⟩] commands work in the usual way. The environment can be nested with at most *"four levels"* and the options can be configured globally using \setenumext command and locally using [⟨key = val⟩] in the environment.

Example with columns=2

1. This text is in the first level.

   (a) This text is in the second level.

      i. This text is in the third level.

         A. This text is in the fourth level.

X This text is in the first level.

⋆ 2. This text is in the first level.

2.2 The environment enumext*

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

Some considerations to take into account for this environment:

- The environment cannot be nested within itself, but it can be nested within enumext and can contain it nested within it.
- Each *"item"* in the environment is placed within a minipage environment whose *width* is stored in the dimension \itemwidthdim that includes labelwith, labelsep plus the *width of the content*.
- You cannot have floating environments like figure or table but \footnote with hyperref support is supported if the footnotehyper package is loaded.

Example with columns=2

1. This text is in the first level. 2. This text is in the first level.

X This text is in the first level. ⋆ 3. This text is in the first level.

2.3 The command \item*

\item*

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

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

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

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

2.3.1 Keys for `\item*`

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

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

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

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

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

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

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

```
   1.  The first           ⋆ 2.  The second            3.  The third      4.  The fourth
 ⋆ 5.  The fifth item is way too long for this and needs three columns    6.  The sixth
   7.  the seventh           X   The eighth item is way too long for this and needs   Z   The ninth
                                 two columns
   8.  The tenth
```

3 The command `\setenumext`

`\setenumext`

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

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

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

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

4 The keyval system

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

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

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

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

4.1 Keys for label and ref

label = {⟨\alph* | \Alph* | \arabic* | \roman* | \Roman* ⟩} default: *by levels*

Sets the ⟨*label*⟩ that will be printed at the *current level*. The default value for the first level of the environments enumext and enumext* are \arabic*., for second level are (\alph*), for third level are \roman*. and for fourth level are \Alph*.. For keyans and keyans* environments the default value is \Alph*).

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

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

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

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

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

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

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*⟩].

4.2 Keys for spaces

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

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

4.2.1 Vertical spaces

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

4.2.2 Horizontal spaces

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

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

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

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

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

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

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

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

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

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

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

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

4.3 Keys for add code

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

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

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

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

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

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

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

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

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

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

4.4 Keys for start, series and resume

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

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

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

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

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

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

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

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

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

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

4.5 Keys for multicols

`columns = {⟨integer⟩}` default: *1*

Set the *number of columns* to be used by the `multicols` environment within the environment. The value must be a positive integer less than or equal to `10`.

`columns-sep = {⟨rigid length⟩}` default: *by level*

Set the *space between* columns used by the `multicols` environment within the environment. Internally sets the value of `\columnsep`, by default its value is equal to the sum of the values set in the keys `labelwidth` and `labelsep` of the current level.

The `\footnote{⟨text⟩}` command in the nested levels of `multicols` will not work as expected, prefer the use of `\footnotemark[⟨number⟩]` inside the environment and `\footnotetext[⟨number⟩]{⟨text⟩}` outside the environment or via the `after` key.

4.6 Keys for minipage

`mini-env = {⟨rigid length⟩}` default: *not used*

Sets the *width* of the `minipage` environment on the *"right side"*. This value added to the value set by the `mini-sep` key to determines the *width* of the `minipage` environment on the *"left side"*, taking `\linewidth` as the maximum reference value.

`mini-sep = {⟨rigid length⟩}` default: *0.3333em*

Sets the *space between* the `minipage` environment on the *"left side"* and the `minipage` environment on the *"right side"*. This separation is applied together with `\hfill`.

4.6.1 The command \miniright

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

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

The `\footnote{⟨text⟩}` command in `minipage` environment will work as usual. If you prefer the footnotes to be numbered (not lowercase) and outside the environment, use `\footnotemark[⟨number⟩]` inside the environment and `\footnotetext[⟨number⟩]{⟨text⟩}` outside the environment or via the `after` key (see §1.3.5 for full support).

4.6.2 The key mini-right

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

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

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

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

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

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

5 The storage system

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

```
\begin{enumext}[save-ans={⟨store name⟩}]
  \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*⟩.

5.1 Keys for storage system

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

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

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

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

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

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

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

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

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

5.1.1 Keys for label and ref

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

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

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

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

5.1.2 Keys for wrap and display

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

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

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

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

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

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

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

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

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

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

5.1.3 Keys for debug and checking

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

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

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

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

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

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

5.2 The command \anskey

\anskey \anskey[⟨keys⟩]{⟨content⟩}

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

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

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

5.2.1 Keys for \anskey

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

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

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

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

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

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

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

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

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

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

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

Example

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

⋆ 1. Text containing our instructions or questions.
  ⋆ first answer
2. Text containing our instructions or questions.
  (a) Question.
    ⋆ second answer
3. Text containing our instructions or questions.
  ⋆ third answer
4. Text containing our instructions or questions.
  ⋆ fourth answer

5.3 The environment anskey*

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

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

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

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

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

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

Example

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

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

⋆ 5. Text containing our instructions or questions.
[5] First answer with verbatim
6. Text containing our instructions or questions.
(a) Question.
[6] second answer
7. Text containing our instructions or questions.
[7] third answer
8. Text containing our instructions or questions.
[8] fourth answer

5.4 The environments keyans and keyans*

keyans
keyans*

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

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

```
\begin{enumext}[save-ans=test]
  \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*).

5.4.1 The \item* in keyans and keyans*

\item*

```
\item*
\item*[⟨content⟩]
```

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

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

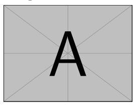
The behavior of \item* in keyans and keyans* environments is NOT the same as in the enumext or enumext* environments.

Example

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

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

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


Some text

5.5 The environment keyanspic

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

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

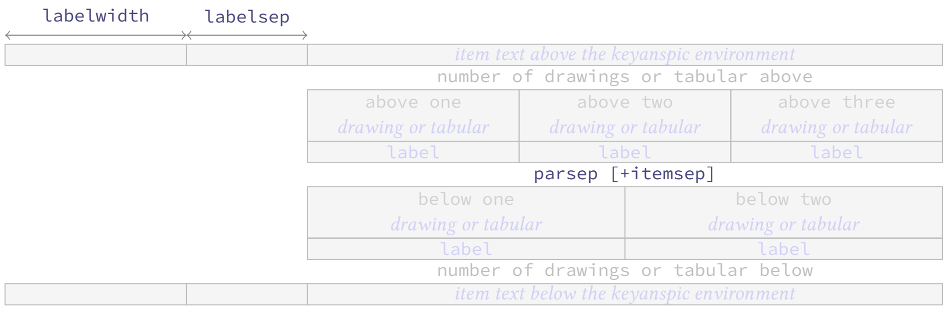


Figure 6: Representation of the keyanspic environment with optional argument [3,2] in enumext.

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

5.5.1 The command \anspic

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

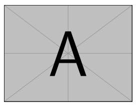
The \anspic command take three arguments, the *starred argument* '*' store the current ⟨*label*⟩ next to the ⟨*content*⟩ (if it is present) in *sequence* and *prop list* {⟨*store name*⟩} set by save-ans key.

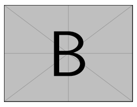
The *starred argument* '*' cannot be separated by spaces '␣' from the command, i.e. \anspic* and the optional argument does *"not support"* verbatim content. By design it is assumed that the *starred argument* '*' will only appear *"once"* within the environment.

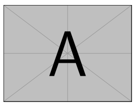
Example

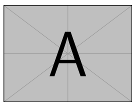
```
\begin{enumext}[save-ans=test,show-ans,nosep]
  \item Question with images.
    \begin{keyanspic}[3,2]
      \anspic{\includegraphics[scale=0.15]{example-image-a}}
      \anspic{\includegraphics[scale=0.15]{example-image-b}}
      \anspic{\includegraphics[scale=0.15]{example-image-a}}
      \anspic{\includegraphics[scale=0.15]{example-image-a}}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

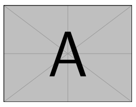
1. Question with images.


A)


B)


C)


D)


⋆ E)[note]

5.6 Printing stored content

5.6.1 The command \getkeyans

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

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

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

5.6.2 The command \printkeyans

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

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

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

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

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

- If we execute `\printkeyans*{⟨store name⟩}` and the *sequence* {⟨*store name*⟩} already contains any `enumext*` environment an error will be returned as we cannot nest.
- If we execute `\printkeyans*{⟨store name⟩}` and the *sequence* {⟨*store name*⟩} contains any `enumext` environments, they will start with the ⟨*keys*⟩ set for the first level unless they are set in the optional argument or `save-key` is used to modify it.
- If we execute `\printkeyans{⟨store name⟩}` and the *sequence* {⟨*store name*⟩} contains any environment `enumext*`, they will start with the ⟨*keys*⟩ set by default unless they are set in the optional argument or `save-key` is used to modify it.

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

Example

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

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

  \item Related to Linux

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

```
\end{enumext}

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

\printkeyans{sample}
```

1. Factor $3x + 3y + 3z$.
   [1] $\boxed{3(x + y + z)}$
2. True False
   (a) LaTeX2e is cool?
      [2] Very True!
3. Related to Linux
   (a) You use linux?
      [3] Yes
   (b) Rate the following package and class
      i. `xsim`
         [4] very good
      ii. `exsheets`
         [5] obsolete

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

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

6 Full examples

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

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

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

Example 1

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

1. La velocità di $1{,}00 \times 10^2$ m/s espressa in km/h è:
   A 36 km/h.
   B 360 km/h.
   C 27,8 km/h.
   D $3{,}60 \times 10^8$ km/h.
2. In fisica nucleare si usa l'angstrom (simbolo: $1\,\text{Å} = 1 \times 10^{-10}$ m) e il fermi o femtometro ($1\,\text{fm} = 1 \times 10^{-15}$ m). Qual è la relazione tra queste due unità di misura?
   A $1\,\text{Å} = 1 \times 10^5$ fm.
   B $1\,\text{Å} = 1 \times 10^{-5}$ fm.
   C $1\,\text{Å} = 1 \times 10^{-15}$ fm.
   D $1\,\text{Å} = 1 \times 10^3$ fm.
3. La velocità di $1{,}00 \times 10^2$ m/s espressa in km/h è:
   A 36 km/h.
   B 360 km/h.
   C 27,8 km/h.
   D $3{,}60 \times 10^8$ km/h.
4. In fisica nucleare si usa l'angstrom (simbolo: $1\,\text{Å} = 1 \times 10^{-10}$ m) e il fermi o femtometro ($1\,\text{fm} = 1 \times 10^{-15}$ m). Qual è la relazione tra queste due unità di misura?
   A $1\,\text{Å} = 1 \times 10^5$ fm.
   B $1\,\text{Å} = 1 \times 10^{-5}$ fm.
   C $1\,\text{Å} = 1 \times 10^{-15}$ fm.
   D $1\,\text{Å} = 1 \times 10^3$ fm.

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

Example 2

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

1. La velocità di $1{,}00 \times 10^2$ m/s espressa in km/h è:
   A 36 km/h.
   ✓ B 360 km/h.
   C 27,8 km/h.
   D $3{,}60 \times 10^8$ km/h.
2. In fisica nucleare si usa l'angstrom (simbolo: $1\,\text{Å} = 1 \times 10^{-10}$ m) e il fermi o femtometro ($1\,\text{fm} = 1 \times 10^{-15}$ m). Qual è la relazione tra queste due unità di misura?
   ✓ A $1\,\text{Å} = 1 \times 10^5$ fm.
   B $1\,\text{Å} = 1 \times 10^{-5}$ fm.
   C $1\,\text{Å} = 1 \times 10^{-15}$ fm.

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

D $1\,\text{Å} = 1 \times 10^3$ fm.

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

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

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

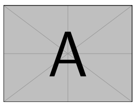
   ✓ A $1\,\text{Å} = 1 \times 10^5$ fm.
   B $1\,\text{Å} = 1 \times 10^{-5}$ fm.
   C $1\,\text{Å} = 1 \times 10^{-15}$ fm.
   D $1\,\text{Å} = 1 \times 10^3$ fm.

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

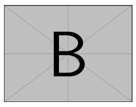
Example 3

A *"simple multiple choice"* test .

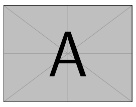
1. First type of questions
   A value
   B correct
   C value
   D value
2. Second type of questions
   I. $2\alpha + 2\delta = 90^\circ$
   II. $\alpha = \delta$
   III. $\angle EDF = 45^\circ$
   A I only
   B II only
   C I and II only
   D I and III only
   E I, II, and III
3. Third type of questions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



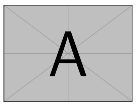
A



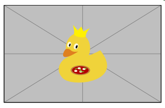
B



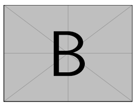
C



D


E

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



Test keys

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

Example 4

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

1 Factor $x^2 - 2x + 1$

2 Factor $3x + 3y + 3z$

The following questions need to be cuaqtified :)

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

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

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

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

Example 5

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

1

Which choice best describes what happens in the passage?

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

2

Which choice best describes what happens in the passage?

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

3

Which choice best describes what happens in the passage?

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

4

Which choice best describes what happens in the passage?

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

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

7 The way of non-enumerated lists

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

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

Fake `itemize` environment

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

- First level item
  - Second level item
    - Third level item
      - Fourth level item
- First level item

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

Fake description environment

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

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

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

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

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

Description indented by label

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

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

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

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

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

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

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

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

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

Description with multi-line labels

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

```
\NewDocumentCommand \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 typessetting a document with LaTeX, that is to say the output does not always look as nice as it should, even if they are only alternatives these must follow a certain order when presented either numerical or presentation, that said handling that using *nested lists* is quite complicated so I do not classify to be implemented.

8 References

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

9 Change history

v1.0 2024-06-13 – First public release.

10 Index of Documentation

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

11 Implementation

The most recent publicly released version of enumext is available at CTAN: https://www.ctan.org/pkg/enumext. While general feedback via email is welcomed, specific bugs or feature requests should be reported through the issue tracker: https://github.com/pablgonz/enumext/issues.

The documentation presented here is far from professional, it contains a lot of obvious information that to the eye of a TEXpert are superfluous, but, after so many years developing this project is the only way to remember what does what.

11.1 General conventions

Variables containing i, ii, iii and iv are associated by level with the enumext environment, variables containing v are associated with the keyans environment, variables containing vi are associated with the keyanspic environment, variables containing vii are associated with the enumext* environment and variables containing viii are associated with the keyans* environment.

To simplify writing and documentation some variables and functions that are common to the different levels of the environments are described using a capital "X".

The temporary function __enumext_tmp:n is used in different parts of the package code for variable creation or execution of other functions that are grouped into this one.

All variables and functions defined in this package are private and are NOT intended to work or be used by another package or module.

11.2 Initial set up

Start the DocStrip guards.

```
⟨*package⟩
```

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

```
⟨@@=enumext⟩
```

11.3 Declaration of the package

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

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

Now declare the enumext package.

```
\ProvidesExplPackage
  {enumext}
  {2024-06-13}
  {1.0}
  {Enumerate exercise sheets}
```

Finally check if the multicol and scontents packages are loaded, if not we load it.

```
\hook_gput_code:nnn {begindocument} {enumext}
  {
    \IfPackageLoadedTF { multicol }
      {
        \msg_info:nnn { enumext } { package-load } { multicol }
      }
      {
        \msg_info:nnn { enumext } { package-not-load } { multicol }
        \RequirePackage{multicol}[2024-05-23]
      }
    \IfPackageLoadedTF { scontents }
      {
        \msg_info:nnn { enumext } { package-load } { scontents }
      }
      {
        \msg_info:nnn { enumext } { package-not-load } { scontents }
        \RequirePackage{scontents}
      }
  }
```

11.4 Definition of variables

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

\l__enumext_level_int
\l__enumext_level_h_int
\l__enumext_anskey_level_int
\l__enumext_keyans_level_int
\l__enumext_keyans_level_h_int
\l__enumext_keyans_pic_level_int

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

```
\int_new:N  \l__enumext_level_int
\int_new:N  \l__enumext_level_h_int
\int_new:N  \l__enumext_anskey_level_int
\int_new:N  \l__enumext_keyans_level_int
\int_new:N  \l__enumext_keyans_level_h_int
\int_new:N  \l__enumext_keyans_pic_level_int
```

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

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

Internal variables used by functions __enumext_is_not_nested:, __enumext_is_on_first_-level: and __enumext_keyans_start_line: (§11.6.1).

```
\bool_new:N \l__enumext_starred_bool
\bool_new:N \g__enumext_starred_bool
\bool_new:N \l__enumext_starred_first_bool
\bool_new:N \l__enumext_standar_bool
\bool_new:N \g__enumext_standar_bool
\bool_new:N \l__enumext_standar_first_bool
\bool_new:N \l__enumext_anskey_env_bool
\bool_new:N \l__enumext_keyans_env_bool
\tl_new:N   \g__enumext_start_line_tl
\tl_new:N   \g__enumext_envir_name_tl
```

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

\l__enumext_counter_i_tl
\l__enumext_counter_ii_tl
\l__enumext_counter_iii_tl
\l__enumext_counter_iv_tl
\l__enumext_counter_v_tl
\l__enumext_counter_vi_tl
\l__enumext_counter_vii_tl
\l__enumext_counter_viii_tl

Variables to store the *"name of the counters"* enumXi, enumXii, enumXiii and enumXiv for enumext environment, enumXv for keyans environment and enumXvi for the keyanspic environment. The counters enumXvii and enumXviii are used by enumext* and keyans* environments.

The initial values of these variables are set by the function __enumext_define_counters:Nn (§11.10) and then modified by the function __enumext_label_style:Nnn used by label key (§11.13).

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

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

\c__enumext_counter_style_tl
\l__enumext_ref_key_arg_tl
\l__enumext_ref_the_count_tl
\l__enumext_the_counter_X_tl
\l__enumext_renew_the_count_X_tl

Internal variables used by ref key (§11.13).

```
\tl_const:Nn \c__enumext_counter_style_tl
  { { arabic } { roman } { Roman } { alph } { Alph } }
\tl_new:N \l__enumext_ref_key_arg_tl
\tl_new:N \l__enumext_ref_the_count_tl
\cs_set_protected:Npn \__enumext_tmp:n #1
  {
    \tl_new:c  { l__enumext_renew_the_count_#1_tl }
    \tl_new:c  { l__enumext_the_counter_#1_tl }
    \tl_set:ce { l__enumext_the_counter_#1_tl } { \exp_not:c { theenumX#1 } }
  }
\clist_map_inline:nn { i, ii, iii, iv, v, vi, vii, viii } { \__enumext_tmp:n {#1} }
```

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

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

Internal variables used by resume, resume* and series keys (§11.24).

```
\int_new:N  \g__enumext_resume_int
\int_new:N  \g__enumext_resume_vii_int
\tl_new:N   \l__enumext_resume_name_tl
\bool_new:N \l__enumext_resume_active_bool
\tl_new:N   \g__enumext_standar_series_tl
\tl_new:N   \g__enumext_starred_series_tl
```

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

\l__enumext_current_widest_dim
\g__enumext_counter_styles_tl
\g__enumext_widest_label_tl
\l__enumext_label_width_by_box

The variable \l__enumext_current_widest_dim stores the current label width, the variable \g__-enumext_counter_styles_tl stores the default ⟨*label style*⟩ and the variable \g__enumext_widest_-label_tl the label width. These variables are used by widest (§11.14) and label (§11.12) keys.

```
\dim_new:N \l__enumext_current_widest_dim
\tl_new:N  \g__enumext_counter_styles_tl
\tl_new:N  \g__enumext_widest_label_tl
\box_new:N \l__enumext_label_width_by_box
```

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

\l__enumext_leftmargin_tmp_X_bool
\l__enumext_leftmargin_tmp_X_dim
\l__enumext_leftmargin_X_dim
\l__enumext_itemindent_X_dim

The boolean variable \l__enumext_leftmargin_tmp_X_bool and the dimensional variable \l__-enumext_leftmargin_tmp_X_dim are used by the list-indent key (§11.17). The variables \l__-enumext_leftmargin_X_dim and \l__enumext_itemindent_X_dim are used and set by the function __enumext_calc_hspace:NNNNNNNNNNNN (§11.37.1).

```
\cs_set_protected:Npn \__enumext_tmp:n #1
  {
    \bool_new:c { l__enumext_leftmargin_tmp_#1_bool }
    \dim_new:c  { l__enumext_leftmargin_tmp_#1_dim }
    \dim_new:c  { l__enumext_leftmargin_#1_dim     }
    \dim_new:c  { l__enumext_itemindent_#1_dim     }
  }
\clist_map_inline:nn { i, ii, iii, iv, v, vi, vii, viii } { \__enumext_tmp:n {#1} }
```

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

\l__enumext_multicols_above_X_skip
\l__enumext_multicols_below_X_skip

Internal variables used by columns key §11.21).

```
\cs_set_protected:Npn \__enumext_tmp:n #1
  {
    \skip_new:c  { l__enumext_multicols_above_#1_skip }
    \skip_new:c  { l__enumext_multicols_below_#1_skip }
  }
\clist_map_inline:nn { i, ii, iii, iv, v } { \__enumext_tmp:n {#1} }
```

(*End of definition for* \l__enumext_multicols_above_X_skip *and* \l__enumext_multicols_below_X_skip.)

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

Internal variables used by \miniright command (§11.22.4) and the keys mini-right, mini-right*, mini-env and mini-sep (§11.20, §11.22).

```
\int_new:N  \g__enumext_minipage_stat_int
\skip_new:N \l__enumext_minipage_left_skip
\skip_new:N \l__enumext_minipage_right_skip
\skip_new:N \l__enumext_minipage_after_skip
\skip_new:N \g__enumext_minipage_right_skip
\skip_new:N \g__enumext_minipage_after_skip
\cs_set_protected:Npn \__enumext_tmp:n #1
  {
    \dim_new:c  { l__enumext_minipage_left_#1_dim    }
    \bool_new:c { l__enumext_minipage_active_#1_bool }
  }
\clist_map_inline:nn { i, ii, iii, iv, v, vii, viii } { \__enumext_tmp:n {#1} }
```

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

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

The integer variable \l__enumext_start_X_int are used by the start key (§11.14), the token list \l__enumext_fake_item_indent_X_tl is used by itemindent key (§11.17.1), the variables \l__-enumext_label_fill_left_X_tl and \l__enumext_label_fill_left_X_tl are used by the align key (§11.12). 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 (§11.19).

```
\cs_set_protected:Npn \__enumext_tmp:n #1
  {
    \bool_new:c { l__enumext_wrap_label_#1_bool     }
    \bool_new:c { l__enumext_wrap_label_opt_#1_bool }
    \int_new:c  { l__enumext_start_#1_int           }
    \tl_new:c   { l__enumext_fake_item_indent_#1_tl }
    \tl_new:c   { l__enumext_label_fill_left_#1_tl  }
    \tl_new:c   { l__enumext_label_fill_right_#1_tl }
    \bool_new:c { l__enumext_vspace_a_star_#1_bool  }
    \bool_new:c { l__enumext_vspace_b_star_#1_bool  }
  }
\clist_map_inline:nn { i, ii, iii, iv, v, vii, viii } { \__enumext_tmp:n {#1} }
```

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

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

The variable \l__enumext_store_active_bool setting by save-ans key (§11.25.1) 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 (§11.29) 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* (§11.30).

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* (§11.35.2) and \anspic* (§11.40.1) for the keyans, keyans* and keyanspic environments.

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

```
\bool_new:N \l__enumext_store_active_bool
\tl_new:N   \l__enumext_store_name_tl
\tl_new:N   \g__enumext_store_name_tl
\tl_new:N   \l__enumext_store_anskey_arg_tl
\tl_new:N   \l__enumext_store_anskey_env_tl
\tl_new:N   \l__enumext_store_anskey_opt_tl
\tl_new:N   \l__enumext_store_current_label_tl
\tl_new:N   \l__enumext_store_current_opt_arg_tl
\tl_new:N   \l__enumext_store_current_label_tmp_tl
```

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

\l__enumext_setkey_tmpa_tl
\l__enumext_setkey_tmpb_tl
\l__enumext_setkey_tmpa_int
\l__enumext_setkey_tmpa_seq
\l__enumext_setkey_tmpb_seq

Internal variables used by the command \setenumext (§11.46).

```
\tl_new:N  \l__enumext_setkey_tmpa_tl
\tl_new:N  \l__enumext_setkey_tmpb_tl
\int_new:N \l__enumext_setkey_tmpa_int
\seq_new:N \l__enumext_setkey_tmpa_seq
\seq_new:N \l__enumext_setkey_tmpb_seq
```

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

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

Internal variables used by command \printkeyans (§11.45), show-pos key (§11.26), item-sym* key (§11.33), save-key key (§11.26.2) and *"storage level system"*.

```
\tl_new:N  \l__enumext_print_keyans_starred_tl
\str_new:N \l__enumext_mark_position_str
\tl_new:N  \g__enumext_item_symbol_tl
\cs_set_protected:Npn \__enumext_tmp:n #1
  {
    \tl_new:c   { l__enumext_print_keyans_#1_tl        }
    \tl_new:c   { l__enumext_store_save_key_#1_tl      }
    \bool_new:c { l__enumext_store_save_key_#1_bool    }
    \bool_new:c { l__enumext_store_upper_level_#1_bool }
  }
\clist_map_inline:nn { i, ii, iii, iv, vii } { \__enumext_tmp:n {#1} }
```

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

\l__enumext_keyans_pic_body_seq
\l__enumext_keyans_pic_width_dim
\l__enumext_keyans_pic_above_int
\l__enumext_keyans_pic_below_int
\l__enumext_keyans_pic_above_skip

Internal variables used by keyanspic environment (§11.40.2).

```
\seq_new:N  \l__enumext_keyans_pic_body_seq
\dim_new:N  \l__enumext_keyans_pic_width_dim
\int_new:N  \l__enumext_keyans_pic_above_int
\int_new:N  \l__enumext_keyans_pic_below_int
\skip_new:N \l__enumext_keyans_pic_above_skip
```

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

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

Internal variables used by *"internal check answer"* mechanism (§11.25.3) 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.

```
\bool_new:N \l__enumext_check_answers_bool
\bool_new:N \g__enumext_check_ans_key_bool
\tl_new:N   \l__enumext_check_start_line_env_tl
\int_new:N  \g__enumext_check_starred_cmd_int
\int_new:N  \g__enumext_item_anskey_int
```

```
\int_new:N  \g__enumext_item_number_int
\bool_new:N \l__enumext_item_number_bool
\int_new:N  \g__enumext_item_answer_diff_int
```

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

`\l__enumext_hyperref_bool`
`\l__enumext_footnotes_key_bool`

The boolean variable `\l__enumext_hyperref_bool` will determine if the `hyperref` package is present or load in memory (§11.9). The boolean variable `\l__enumext_footnotes_key_bool` determine if `hyperref` is load with key `hyperfootnotes=true`.

```
\bool_new:N \l__enumext_hyperref_bool
\bool_new:N \l__enumext_footnotes_key_bool
```

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

`\l__enumext_newlabel_arg_one_tl`
`\l__enumext_newlabel_arg_two_tl`
`\l__enumext_write_aux_file_tl`
`\l__enumext_label_copy_X_tl`

Internal variables used by `save-ref` key (§11.26). The variables `\l__enumext_label_copy_X_tl` correspond to temporary copies of the ⟨*labels*⟩ defined by level on which operations will be performed.

The variables `\l__enumext_newlabel_arg_one_tl` and `\l__enumext_newlabel_arg_two_tl` will be used to form the arguments passed to the function `\__enumext_newlabel:nn` (§11.9) and the variable `\l__enumext_write_aux_file_tl` will be in charge of executing the writing code in the `.aux` file.

```
\tl_new:N \l__enumext_newlabel_arg_one_tl
\tl_new:N \l__enumext_newlabel_arg_two_tl
\tl_new:N \l__enumext_write_aux_file_tl
\cs_set_protected:Npn \__enumext_tmp:n #1
  {
    \tl_new:c { l__enumext_label_copy_#1_tl }
  }
\clist_map_inline:nn { i, ii, iii, iv, v, vi, vii, viii } { \__enumext_tmp:n {#1} }
```

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

`\g__enumext_footnote_int`
`\g__enumext_footnote_arg_seq`
`\g__enumext_footnote_int_seq`

Internal variables used for redefinition of `\footnote` (§11.34).

```
\int_new:N \g__enumext_footnote_int
\seq_new:N \g__enumext_footnote_arg_seq
\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.

```
\cs_set_protected:Npn \__enumext_tmp:n #1
  {
    \bool_new:c { l__enumext_item_starred_#1_bool     }
    \int_new:c  { l__enumext_item_column_pos_#1_int   }
    \int_new:c  { g__enumext_item_count_all_#1_int    }
    \int_new:c  { l__enumext_joined_item_#1_int       }
    \int_new:c  { l__enumext_joined_item_aux_#1_int   }
    \int_new:c  { l__enumext_tmpa_#1_int              }
    \dim_new:c  { l__enumext_tmpa_#1_dim              }
    \box_new:c  { l__enumext_item_text_#1_box         }
    \dim_new:c  { l__enumext_joined_width_#1_dim      }
    \dim_new:c  { l__enumext_item_width_#1_dim        }
    \tl_new:c   { g__enumext_item_symbol_aux_#1_tl    }
    \str_new:c  { l__enumext_align_label_#1_str       }
    \bool_new:c { g__enumext_minipage_active_#1_bool }
    \box_new:c  { l__enumext_miniright_code_#1_box    }
    \bool_new:c { g__enumext_minipage_center_#1_bool }
    \dim_new:c  { g__enumext_minipage_right_#1_dim    }
    \skip_new:c { g__enumext_minipage_right_#1_skip   }
  }
\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`.

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

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

11.5 Public dimension

The package enumext only provides a single public dimension \itemwidthdim 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.

```
\dim_zero_new:N \itemwidthdim
```

11.6 Some utility functions

__enumext_at_begin_document:n

A internal *"hook"* function used for copying plain list and minipage environments definition and hyperref detection.

```
\cs_new_protected:Npn \__enumext_at_begin_document:n #1
  {
    \hook_gput_code:nnn {begindocument} {enumext} { #1 }
  }
```

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

__enumext_after_env:nn
__enumext_before_env:nn

A internal *"hook"* functions for execute code mini-right and mini-right* keys outside the enumext* and keyans* environments and print check-ans outside the enumext and enumext* environments.

```
\cs_new_protected:Npn \__enumext_after_env:nn #1 #2
  {
    \hook_gput_code:nnn {env/#1/after} {enumext} {#2}
  }
\cs_new_protected:Npn \__enumext_before_env:nn #1 #2
  {
    \hook_gput_code:nnn {env/#1/before} {enumext} {#2}
  }
```

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

__enumext_level:

Function for check current level in enumext.

```
\cs_new:Nn \__enumext_level:
  {
    \int_to_roman:n { \l__enumext_level_int }
  }
```

(*End of definition for* __enumext_level:.)

__enumext_if_is_int:nT
__enumext_if_is_int:nF
__enumext_if_is_int:nTF

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

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

(*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.

```
\cs_new_protected:Nn \__enumext_regex_counter_style:
  {
    \tl_map_inline:Nn \c__enumext_counter_style_tl
      {
        \regex_replace_once:nnN { \c{##1}\* }
          { \c{##1}\cB{\u{l__enumext_ref_the_count_tl}\cE} } \l__enumext_ref_key_arg_tl
      }
  }
```

(*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.

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

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

11.6.1 Utilities for environments and levels

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

```
\cs_new_protected:Nn \__enumext_is_not_nested:
  {
    \str_case:en { \@currenvir }
      {
        {enumext}
          {
            \bool_lazy_and:nnT
              { \bool_not_p:n { \g__enumext_standar_bool } }
              { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
              {
                \bool_gset_true:N \g__enumext_standar_bool
              }
          }
        {enumext*}
          {
            \bool_lazy_and:nnT
              { \bool_not_p:n { \g__enumext_starred_bool } }
              { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
              {
                \bool_gset_true:N \g__enumext_starred_bool
              }
          }
      }
  }
```

The function `\__enumext_is_on_first_level:` will set the variables `\l__enumext_standar_first_bool` (§11.25.1), `\l__enumext_starred_first_bool` (§11.25.1) and `\l__enumext_anskey_env_bool` (§11.30) 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.

```
\cs_new_protected:Nn \__enumext_is_on_first_level:
  {
    \bool_lazy_all:nT
      {
        { \bool_if_p:N \g__enumext_standar_bool }
        { \int_compare_p:nNn { \l__enumext_level_int } = { 1 } }
        { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
      }
      {
        \bool_set_true:N \l__enumext_standar_first_bool
        \bool_set_true:N \l__enumext_anskey_env_bool
        \tl_gset:Nn \g__enumext_envir_name_tl { enumext }
        \tl_gset:Ne \g__enumext_start_line_tl
          {
            on ~ line ~ \exp_not:V \inputlineno
          }
      }
    \bool_lazy_all:nT
      {
        { \bool_if_p:N \g__enumext_starred_bool }
        { \int_compare_p:nNn { \l__enumext_level_h_int } = { 1 } }
        { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
      }
      {
        \bool_set_true:N \l__enumext_starred_first_bool
        \bool_set_true:N \l__enumext_anskey_env_bool
        \tl_gset:Nn \g__enumext_envir_name_tl { enumext* }
        \tl_gset:Ne \g__enumext_start_line_tl
          {
            on ~ line ~ \exp_not:V \inputlineno
          }
      }
  }
```

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

`\__enumext_keyans_start_line:`

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

```
\cs_new_protected:Nn \__enumext_keyans_start_line:
  {
    \str_case:en { \@currenvir }
      {
        {keyans}
          {
            \tl_set:Ne \l__enumext_check_start_line_env_tl
              {
                in ~ 'keyans' ~ start ~ on ~ line ~ \exp_not:V \inputlineno
              }
          }
        {keyans*}
          {
            \tl_set:Ne \l__enumext_check_start_line_env_tl
              {
                in ~ 'keyans*' ~ start ~ on ~ line ~ \exp_not:V \inputlineno
              }
          }
        {keyanspic}
          {
            \tl_set:Ne \l__enumext_check_start_line_env_tl
              {
                in ~ 'keyanspic' ~ start ~ on ~ line ~ \exp_not:V \inputlineno
              }
          }
      }
  }
```

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

11.6.2 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.

```
\cs_new_protected:Nn \__enumext_reset_global_vars:
  {
    \__enumext_reset_global_int:
    \__enumext_reset_global_bool:
    \__enumext_reset_global_tl:
  }
\cs_new_protected:Nn \__enumext_reset_global_int:
  {
    \int_gzero:N \g__enumext_item_number_int
    \int_gzero:N \g__enumext_item_anskey_int
    \int_gzero:N \g__enumext_item_answer_diff_int
  }
\cs_new_protected:Nn \__enumext_reset_global_bool:
  {
    \bool_gset_false:N \g__enumext_check_ans_key_bool
    \bool_gset_false:N \g__enumext_standar_bool
    \bool_gset_false:N \g__enumext_starred_bool
  }
\cs_new_protected:Nn \__enumext_reset_global_tl:
  {
    \tl_gclear:N \g__enumext_store_name_tl
    \tl_gclear:N \g__enumext_start_line_tl
    \tl_gclear:N \g__enumext_envir_name_tl
  }
```

(*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.

```
\cs_new_protected:Nn \__enumext_log_global_vars:
  {
    \msg_log:nneeee { enumext } { prop-seq-int-hook }
      { \g__enumext_store_name_tl }
      { \prop_count:c { g__enumext_ \g__enumext_store_name_tl _prop } }
      { \seq_count:c { g__enumext_ \g__enumext_store_name_tl _seq } }
      { \int_use:c { g__enumext_resume_ \g__enumext_store_name_tl _int } }
  }
```

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.

```
\cs_new_protected:Nn \__enumext_log_answer_vars:
  {
    \msg_log:nneee { enumext } { item-answer-hook }
      { \int_use:N \g__enumext_item_number_int }
      { \int_use:N \g__enumext_item_anskey_int }
      { \int_eval:n { \g__enumext_item_number_int - \g__enumext_item_anskey_int} }
  }
```

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

11.7 Copying `list` and `minipage` environments

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

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

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

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

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

```
\__enumext_at_begin_document:n
  {
    \cs_new_eq:NN \__enumext_start_list:nn \list
    \cs_new_eq:NN \__enumext_stop_list: \endlist
    \cs_new_eq:NN \__enumext_item_std:w \item
  }
```

(*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.

```
\__enumext_at_begin_document:n
  {
    \cs_new_eq:NN \__enumext_minipage:w \minipage
    \cs_new_eq:NN \__enumext_endminipage: \endminipage
  }
```

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

11.8 The internal minipage environment

__enumext_internal_mini_page:
__enumext_mini_env*

The function __enumext_internal_mini_page: creates a internal __enumext_mini_env* environment (*custom version* of minipage) setting the \if@minipage switch to *"false"* to allow spaces at the *"above"* of the environment, plus we will add \vspace{0pt} to maintain alignment on *"top"*. This environment will be used internally by the mini-env key, it is not documented in the user interface and is for internal use only. This function is passed to the function __enumext_safe_exec: in the enumext environment definition (§11.38) and __enumext_safe_exec_vii: in the enumext* environment definition (§11.42)

```
\cs_new_protected:Nn \__enumext_internal_mini_page:
  {
    \int_compare:nNnT { \l__enumext_level_int } = { 0 }
      {
        \DeclareDocumentEnvironment{__enumext_mini_env*}{ m }
          {
            \__enumext_minipage:w [ t ] { ##1 }
              \legacy_if_gset_false:n { @minipage }
              \vspace { 0pt }
          }
          { \__enumext_endminipage: }
      }
  }
```

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

11.9 Compatibility with hyperref and footnotehyper

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

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

```
\cs_new_protected:Nn \__enumext_after_hyperref:
  {
    \IfPackageLoadedTF { hyperref }
      {
        \msg_info:nnn { enumext } { package-load } { hyperref }
        \bool_set_true:N \l__enumext_hyperref_bool
        \IfHyperBoolean{hyperfootnotes}
          {
            \typeout{hyperfootnotes=true}
            \bool_set_true:N \l__enumext_footnotes_key_bool
          }
          { \typeout{hyperfootnotes=false} }
      }
      {  }
```

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.

```
   \bool_if:NT \l__enumext_footnotes_key_bool
     {
       \IfPackageLoadedTF { footnotehyper }
         {
           \msg_info:nnn { enumext } { package-load } { footnotehyper }
         }
         {
           \typeout{No ~ footnotehyper ~ load}
           \typeout{Load ~ and  ~ use  ~ \string\makesavenoteenv{enumext*}}
           \bool_set_false:N \l__enumext_footnotes_key_bool
         }
     }
```

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.

```
    \bool_if:NTF \l__enumext_hyperref_bool
      {
        \cs_new_eq:NN \__enumext_hypertarget:nn \hypertarget
        \cs_new_eq:NN \__enumext_phantomsection: \phantomsection
      }
      {
        \cs_new_eq:NN \__enumext_hypertarget:nn \use_none:nn
        \cs_new_eq:NN \__enumext_phantomsection: \prg_do_nothing:
      }
  }
```

(*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.

```
\cs_new_protected:Npn \__enumext_newlabel:nn #1 #2
  {
    \protected@write \@auxout { }
      {
        \token_to_str:N \newlabel {#1}
          {
            {#2}
            \bool_if:NT \l__enumext_hyperref_bool
              { { \thepage } {#2} {#1} }
            { }
          }
      }
    \__enumext_hypertarget:nn {#1} { }
    \__enumext_phantomsection:
  }
```

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

11.10 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.

```
\cs_new_protected:Npn \__enumext_define_counters:Nn #1 #2
  {
    \cs_if_exist:cTF { c@ #2 }
      { \msg_fatal:nnn { enumext } { counters }{ #2 } }
      {
        \tl_set:Nn #1 { #2 }
        \newcounter { #2 }
      }
  }
```

(*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.

```
\__enumext_define_counters:Nn \l__enumext_counter_i_tl    { enumXi    }
\__enumext_define_counters:Nn \l__enumext_counter_ii_tl   { enumXii   }
\__enumext_define_counters:Nn \l__enumext_counter_iii_tl  { enumXiii  }
\__enumext_define_counters:Nn \l__enumext_counter_iv_tl   { enumXiv   }
\__enumext_define_counters:Nn \l__enumext_counter_v_tl    { enumXv    }
\__enumext_define_counters:Nn \l__enumext_counter_vi_tl   { enumXvi   }
\__enumext_define_counters:Nn \l__enumext_counter_vii_tl  { enumXvii  }
\__enumext_define_counters:Nn \l__enumext_counter_viii_tl { enumXviii }
```

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

11.11 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.

```
\cs_new_protected:Npn \__enumext_register_counter_style:Nn #1 #2
  {
    \tl_const:cn { c__enumext_widest_ \cs_to_str:N #1 _tl } {#2}
    \tl_gput_right:Nn \g__enumext_counter_styles_tl {#1}
  }
\__enumext_register_counter_style:Nn \arabic { 0 }
\__enumext_register_counter_style:Nn \Alph   { M }
\__enumext_register_counter_style:Nn \alph   { m }
\__enumext_register_counter_style:Nn \Roman  { VIII }
\__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.

```
\cs_new_protected:Npn \__enumext_label_width_by_box:Nn #1 #2
  {
    \hbox_set:Nn \l__enumext_label_width_by_box {#2}
    \dim_set:Nn #1 { \box_wd:N \l__enumext_label_width_by_box }
  }
\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.

```
\cs_new_protected:Npn \__enumext_label_style:Nnn #1 #2 #3
  {
    \tl_clear_new:N #1
    \tl_put_right:Ne #1 { \tl_trim_spaces:n {#3} }
    \tl_gset_eq:NN \g__enumext_widest_label_tl #1
    \tl_map_inline:Nn \g__enumext_counter_styles_tl
      {
        \tl_replace_all:Nne #1 { ##1* } { \exp_not:N ##1 {#2} }
        \tl_greplace_all:Nne \g__enumext_widest_label_tl { ##1* }
          { \tl_use:c { c__enumext_widest_ \cs_to_str:N ##1 _tl } }
      }
    \__enumext_label_width_by_box:Nn \l__enumext_current_widest_dim
      { \tl_use:N \g__enumext_widest_label_tl }
    \tl_set_eq:cN { the #2 } #1
  }
\cs_generate_variant:Nn \__enumext_label_style:Nnn { cvn }
```

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

11.12 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.

```
\cs_set_protected:Npn \__enumext_tmp:nn #1 #2
  {
    \keys_define:nn { enumext / #1 }
      {
        font        .tl_set:c   = { l__enumext_label_font_style_#2_tl },
        font        .value_required:n = true,
        labelsep    .dim_set:c  = { l__enumext_labelsep_#2_dim },
        labelsep    .initial:n  = {0.3333em},
        labelsep    .value_required:n = true,
        labelwidth  .dim_set:c  = { l__enumext_labelwidth_#2_dim },
        labelwidth  .value_required:n = true,
        wrap-label  .cs_set_protected:cp = { __enumext_wrapper_label_#2:n } ##1,
        wrap-label  .initial:n  = {##1},
        wrap-label  .value_required:n = true,
        wrap-label* .code:n = {
                                  \bool_set_true:c { l__enumext_wrap_label_opt_#2_bool }
                                  \keys_set:nn { enumext / #1 } { wrap-label = {##1} }
                                },
        wrap-label* .value_required:n = true,
      }
  }
\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.

```
\cs_set_protected:Npn \__enumext_tmp:nn #1 #2
  {
    \keys_define:nn { enumext / #1 }
      {
        align .choice:,
        align / left    .code:n =
                          {
                            \tl_clear:c { l__enumext_label_fill_left_#2_tl  }
                            \tl_set:cn  { l__enumext_label_fill_right_#2_tl } { \hfill }
                          },
        align / right   .code:n =
                          {
                            \tl_set:cn  { l__enumext_label_fill_left_#2_tl  } { \hfill }
                            \tl_clear:c { l__enumext_label_fill_right_#2_tl }
                          },
        align / center  .code:n =
                          {
                            \tl_set:cn { l__enumext_label_fill_left_#2_tl  } { \hfill }
                            \tl_set:cn { l__enumext_label_fill_right_#2_tl } { \hfill }
                          },
        align / unknown .code:n =
                          \msg_error:nneee { enumext } { unknown-choice }
                            { align } { left, ~ right, ~  center } { \exp_not:n {##1} },
        align .initial:n  = left,
        align .value_required:n  = true,
      }
  }
\clist_map_inline:nn
  {
    {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {keyans}{v}
  }
  { \__enumext_tmp:nn #1 }

\cs_set_protected:Npn \__enumext_tmp:nn #1 #2
  {
    \keys_define:nn { enumext / #1 }
      {
        align .choice:,
        align / left    .code:n = \str_set:cn { l__enumext_align_label_#2_str } { l },
        align / right   .code:n = \str_set:cn { l__enumext_align_label_#2_str } { r },
        align / center  .code:n = \str_set:cn { l__enumext_align_label_#2_str } { c },
        align / unknown .code:n =
                          \msg_error:nneee { enumext } { unknown-choice }
                            { align } { left, ~ right, ~  center } { \exp_not:n {##1} },
        align .initial:n = left,
        align .value_required:n = true,
      }
  }
\clist_map_inline:nn { {enumext*}{vii}, {keyans*}{viii} } { \__enumext_tmp:nn #1 }
```

(*End of definition for* `align`.)

11.13 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.

11.13.1 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.

```
\cs_set_protected:Npn \__enumext_tmp:nnn #1 #2 #3
  {
    \keys_define:nn { enumext / #1 }
      {
        label .code:n    = {
                            \__enumext_label_style:cvn { l__enumext_label_#2_tl }
                              { l__enumext_counter_#2_tl } {##1}
                            \dim_set_eq:cN  { l__enumext_labelwidth_#2_dim }
                              \l__enumext_current_widest_dim
                          },
        label .initial:n = #3,
        label .value_required:n = true,
        ref   .code:n    = \__enumext_standar_ref:n {##1},
        ref   .value_required:n = true,
      }
  }
\__enumext_tmp:nnn { level-1 } {   i } { \arabic*.}
\__enumext_tmp:nnn { level-2 } {  ii } { (\alph*) }
\__enumext_tmp:nnn { level-3 } { iii } { \roman*. }
\__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.

```
\cs_new_protected:Npn \__enumext_standar_ref:n #1
  {
    \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
    \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
      {
        \msg_error:nnn { enumext } { key-ref-empty } { enumext }
      }
      {
        \tl_set_eq:Nc
          \l__enumext_ref_the_count_tl { l__enumext_counter_ \__enumext_level: _tl }
        \__enumext_regex_counter_style:
        \tl_set_eq:Nc
          \l__enumext_ref_the_count_tl { l__enumext_the_counter_ \__enumext_level: _tl }
        \tl_put_right:ce { l__enumext_renew_the_count_ \__enumext_level: _tl }
          {
            \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl }
              { \exp_not:V \l__enumext_ref_key_arg_tl }
          }
      }
  }
```

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

```
\cs_new_protected:Nn \__enumext_standar_ref:
  {
    \tl_if_empty:cF { l__enumext_renew_the_count_ \__enumext_level: _tl }
      {
        \tl_use:c { l__enumext_renew_the_count_ \__enumext_level: _tl }
      }
  }
```

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

11.13.2 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.

```
\cs_set_protected:Npn \__enumext_tmp:nnn #1 #2 #3
  {
    \keys_define:nn { enumext / #1 }
      {
        label .code:n    = {
                             \__enumext_label_style:cvn { l__enumext_label_#2_tl }
                               { l__enumext_counter_#2_tl } {##1}
                             \dim_set_eq:cN  { l__enumext_labelwidth_#2_dim }
                               \l__enumext_current_widest_dim
                           },
        label .initial:n = #3,
        label .value_required:n = true,
        ref   .code:n    = \__enumext_starred_ref:n {##1},
        ref   .value_required:n = true,
      }
  }
\__enumext_tmp:nnn { enumext* } {  vii } { \arabic*.}
\__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.

```
\cs_new_protected:Npn \__enumext_starred_ref:n #1
  {
    \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
    \int_compare:nNnT { \l__enumext_level_h_int } = { 1 }
      {
        \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
          {
            \msg_error:nnn { enumext } { key-ref-empty } { enumext* }
          }
          {
            \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_counter_vii_tl
            \__enumext_regex_counter_style:
            \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_the_counter_vii_tl
            \tl_put_right:Ne \l__enumext_renew_the_count_vii_tl
              {
                \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl }
                  { \exp_not:V \l__enumext_ref_key_arg_tl }
              }
          }
      }
    \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
      {
        \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
          {
            \msg_error:nnn { enumext } { key-ref-empty } { keyans* }
          }
          {
            \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_counter_viii_tl
            \__enumext_regex_counter_style:
            \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_the_counter_viii_tl
            \tl_put_right:Ne \l__enumext_renew_the_count_viii_tl
              {
                \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl }
                  { \exp_not:V \l__enumext_ref_key_arg_tl }
              }
          }
      }
  }
```

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.

```
\cs_new_protected:Nn \__enumext_starred_ref:
  {
    \int_compare:nNnT { \l__enumext_level_h_int } = { 1 }
      {
        \tl_if_empty:NF \l__enumext_renew_the_count_vii_tl
          {
            \tl_use:N \l__enumext_renew_the_count_vii_tl
          }
      }
    \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
      {
        \tl_if_empty:NF \l__enumext_renew_the_count_viii_tl
          {
            \tl_use:N \l__enumext_renew_the_count_viii_tl
          }
      }
  }
```

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

11.13.3 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.

```
\keys_define:nn { enumext / keyans }
  {
    label .code:n     = {
                          \__enumext_label_style:cvn { l__enumext_label_v_tl }
                            { l__enumext_counter_v_tl } {#1}
                          \dim_set_eq:cN  { l__enumext_labelwidth_v_dim }
                            \l__enumext_current_widest_dim
                          \__enumext_label_style:cvn { l__enumext_label_vi_tl }
                             { l__enumext_counter_vi_tl } {#1}
                          \dim_set_eq:cN  { l__enumext_labelwidth_v_dim }
                             \l__enumext_current_widest_dim
                        },
    label .initial:n = \Alph*),
    label .value_required:n = true,
    ref   .code:n     = \__enumext_keyans_ref:n {#1},
    ref   .value_required:n = true,
  }
```

(*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`.

```
\cs_new_protected:Npn \__enumext_keyans_ref:n #1
  {
    \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
    \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
      {
        \msg_error:nnn { enumext } { key-ref-empty } { keyans }
      }
      {
        \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_counter_v_tl
        \__enumext_regex_counter_style:
        \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_the_counter_v_tl
        \tl_put_right:Ne \l__enumext_renew_the_count_v_tl
          {
            \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl }
              { \exp_not:V \l__enumext_ref_key_arg_tl }
          }
      }
  }
```

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

```
\cs_new_protected:Nn \__enumext_keyans_ref:
  {
    \tl_if_empty:NF \l__enumext_renew_the_count_v_tl
      {
        \tl_use:N \l__enumext_renew_the_count_v_tl
      }
  }
```

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

11.14 Setting start and widest keys

__enumext_start_from:NNn
__enumext_start_from:ccn

The function __enumext_start_from:NNn used by the start key take three arguments:
#1 : \l__enumext_label_X_tl
#2 : \l__enumext_start_X_int
#3 : ⟨integer or string⟩
The first argument of this function are the *"counter style"* set by label key, the second argument is returned by the function, the third argument can be an ⟨*integer*⟩ or ⟨*string*⟩ of the form \Alph, \alph, \Roman or \roman. This effectively allows start=A or start=1 to be used.

```
\cs_new_protected:Npn \__enumext_start_from:NNn #1 #2 #3
  {
    \__enumext_if_is_int:nTF { #3 }
       {
         \int_set:Nn #2 {#3}
       }
       {
         \regex_match:nVT { \c{Alph} | \c{alph} } {#1}
           { \int_set:Nn #2 { \int_from_alph:n {#3} } }
         \regex_match:nVT { \c{Roman} | \c{roman} } {#1}
           { \int_set:Nn #2  { \int_from_roman:n {#3} } }
       }
  }
\cs_generate_variant:Nn \__enumext_start_from:NNn { ccn }
```

(*End of definition for* __enumext_start_from:NNn.)

__enumext_widest_from:nNNn
__enumext_widest_from:nccn

The function __enumext_widest_from:nNNn used by the widest key take four arguments:
#1 : The counter associated with the environment level
#2 : \l__enumext_label_X_tl
#3 : \l__enumext_labelwidth_X_dim
#4 : ⟨integer or string⟩
The second and third arguments of this function are the values set by label and labelwidth keys, the four argument can be an ⟨*integer*⟩ or ⟨*string*⟩ of the form \Alph, \alph, \Roman or \roman. The value of the four argument is set temporarily for the identified counter in this point (level), then the value is expanded into a *"box"* and the *"width"* of the *"box"* is returned.

```
\cs_new_protected:Npn \__enumext_widest_from:nNNn #1 #2 #3 #4
  {
    \__enumext_if_is_int:nTF {#4}
      {
        \setcounter{enumX#1} { #4 }
      }
      {
        \regex_match:nVT { \c{Alph} | \c{alph} } {#2}
          { \setcounter{enumX#1} { \int_from_alph:n {#4} } }
        \regex_match:nVT { \c{Roman} | \c{roman} } {#2}
          { \setcounter{enumX#1} { \int_from_roman:n {#4} } }
      }
     \__enumext_label_width_by_box:cv
       { l__enumext_labelwidth_#1_dim } { l__enumext_label_#1_tl }
  }
\cs_generate_variant:Nn \__enumext_widest_from:nNNn { nccn }
```

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

start
widest
\l__enumext_start_X_int

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

```
\cs_set_protected:Npn \__enumext_tmp:nn #1 #2
  {
    \keys_define:nn { enumext / #1 }
      {
        start  .code:n    = {
                              \__enumext_start_from:ccn
                                { l__enumext_label_#2_tl }
                                { l__enumext_start_#2_int } {##1}
                             },
        start  .initial:n = 1,
        widest .code:n    = {
                              \__enumext_widest_from:nccn {#2}
                                { l__enumext_label_#2_tl }
                                { l__enumext_labelwidth_#2_dim } {##1}
                            },
        widest .value_required:n = true,
        start  .value_required:n = true,
      }
  }
\clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }
```

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

11.15 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.

```
\cs_set_protected:Npn \__enumext_tmp:nnnnnn #1 #2 #3 #4 #5 #6
  {
    \keys_define:nn { enumext / #1 }
      {
        topsep    .skip_set:c = { l__enumext_topsep_#2_skip },
        topsep    .initial:n  = {#3},
        topsep    .value_required:n = true,
        partopsep .skip_set:c = { l__enumext_partopsep_#2_skip },
        partopsep .initial:n  = {#4},
        partopsep .value_required:n = true,
        parsep    .skip_set:c = { l__enumext_parsep_#2_skip },
        parsep    .initial:n  = {#5},
        parsep    .value_required:n = true,
        itemsep   .skip_set:c = { l__enumext_itemsep_#2_skip },
        itemsep   .initial:n  = {#6},
        itemsep   .value_required:n = true,
        noitemsep .meta:n     = { itemsep = 0pt, parsep = 0pt },
        noitemsep .value_forbidden:n = true,
        nosep     .meta:n     = {
                                  itemsep = 0pt, parsep= 0pt,
                                  topsep = 0pt, partopsep = 0pt,
                                },
        nosep     .value_forbidden:n = true,
      }
  }
```

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

```
\__enumext_tmp:nnnnnn { level-1 } { i } { 8.0pt plus 2.0pt minus 4.0pt }
  { 2.0pt plus 1.0pt minus 1.0pt } { 4.0pt plus 2.0pt minus 1.0pt }
  { 4.0pt plus 2.0pt minus 1.0pt }
\__enumext_tmp:nnnnnn { level-2 } { ii } { 4.0pt plus 2.0pt minus 1.0pt }
  { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
  { 2.0pt plus 1.0pt minus 1.0pt }
\__enumext_tmp:nnnnnn { level-3 } { iii } { 2.0pt plus 1.0pt minus 1.0pt }
  { 1.0pt minus 1.0pt }{ 0pt }{ 2.0pt plus 1.0pt minus 1.0pt }
\__enumext_tmp:nnnnnn { level-4 } { iv } { 2.0pt plus 1.0pt minus 1.0pt }
  { 1.0pt minus 1.0pt }{ 0pt }{ 2.0pt plus 1.0pt minus 1.0pt }
\__enumext_tmp:nnnnnn { keyans  } { v }{ 4.0pt plus 2.0pt minus 1.0pt }
  { 2.0pt plus 1.0pt minus 1.0pt }{ 2.0pt plus 1.0pt minus 1.0pt }
  { 2.0pt plus 1.0pt minus 1.0pt }
\__enumext_tmp:nnnnnn { enumext* } { vii } { 8.0pt plus 2.0pt minus 4.0pt }
  { 2.0pt plus 1.0pt minus 1.0pt } { 4.0pt plus 2.0pt minus 1.0pt }
  { 4.0pt plus 2.0pt minus 1.0pt }
\__enumext_tmp:nnnnnn { keyans* } { viii } { 4.0pt plus 2.0pt minus 1.0pt }
  { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
  { 2.0pt plus 1.0pt minus 1.0pt }
```

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

11.16 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*`.

```
\cs_set_protected:Npn \__enumext_tmp:n #1
  {
    \keys_define:nn { enumext / #1 }
```

```
      {
        base-fix .bool_set:N = \l__enumext_base_line_fix_bool,
        base-fix .initial:n  = false,
        base-fix .value_forbidden:n = true,
      }
  }
\clist_map_inline:nn { level-1, enumext* } { \__enumext_tmp:n {#1} }
```

The function `\__enumext_nested_base_line_fix:` will be in charge of applying the baseline correction and adjusting the ⟨*keys*⟩. This function is passed to the function `\__enumext_parse_keys:n` in the `enumext` environment definition (§11.38) and to the function `\__enumext_parse_keys_vii:n` in the `enumext*` environment definition (§11.42)

```
\cs_new_protected:Nn \__enumext_nested_base_line_fix:
  {
    \bool_lazy_and:nnT
      { \bool_if_p:N \l__enumext_standar_first_bool }
      { \bool_if_p:N \l__enumext_base_line_fix_bool }
      {
        \mode_leave_vertical:
        \vspace { -\baselineskip }
        \keys_set:nn { enumext / level-1 }
          {
            topsep = 0pt, above = 0pt, above* = 0pt,
          }
      }
    \bool_lazy_and:nnT
      { \bool_if_p:N \l__enumext_starred_first_bool }
      { \bool_if_p:N \l__enumext_base_line_fix_bool }
      {
        \mode_leave_vertical:
        \vspace { -\baselineskip }
        \keys_set:nn { enumext / enumext* }
          {
            topsep = 0pt, above = 0pt, above* = 0pt,
          }
      }
    \bool_set_false:N \l__enumext_base_line_fix_bool
  }
```

This key is enabled by default in the command `\printkeyans` (§11.45).

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

11.17 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.

```
\cs_set_protected:Npn \__enumext_tmp:nn #1 #2
  {
    \keys_define:nn { enumext / #1 }
      {
        itemindent    .dim_set:c = { l__enumext_fake_item_indent_#2_dim },
        itemindent    .value_required:n = true,
        rightmargin   .dim_set:c = { l__enumext_rightmargin_#2_dim },
        rightmargin   .value_required:n = true,
        listparindent .dim_set:c = { l__enumext_listparindent_#2_dim },
        listparindent .value_required:n = true,
        list-offset   .dim_set:c = { l__enumext_listoffset_#2_dim },
        list-offset   .value_required:n = true,
        list-indent   .code:n    =
                        \bool_set_true:c { l__enumext_leftmargin_tmp_#2_bool }
                        \dim_set:cn { l__enumext_leftmargin_tmp_#2_dim } {##1},
        list-indent   .value_required:n = true,
      }
  }
\clist_map_inline:nn
  {
    {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {keyans}{v}
  }
  { \__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.

```
\cs_set_protected:Npn \__enumext_tmp:nn #1 #2
  {
    \keys_define:nn { enumext / #1 }
      {
        itemindent    .dim_set:c = { l__enumext_fake_item_indent_#2_dim },
        itemindent    .value_required:n = true,
        rightmargin   .dim_set:c = { l__enumext_rightmargin_#2_dim },
        rightmargin   .value_required:n = true,
        listparindent .dim_set:c = { l__enumext_listparindent_#2_dim },
        listparindent .value_required:n = true,
        list-offset   .dim_set:c = { l__enumext_listoffset_#2_dim },
        list-offset   .value_required:n = true,
        list-indent   .meta:n    = { list-offset = ##1 },
        list-indent   .value_required:n = true,
      }
  }
\clist_map_inline:nn
  {
    {enumext*}{vii}, {keyans*}{viii}
  }
  { \__enumext_tmp:nn #1 }
```

11.17.1 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.

```
\cs_set_protected:Nn \__enumext_fake_item:
  {
    \dim_compare:nNnT
      { \dim_use:c { l__enumext_fake_item_indent_ \__enumext_level: _dim } }
      >
      { \c_zero_dim }
      {
        \tl_set:ce { l__enumext_fake_item_indent_ \__enumext_level: _tl }
          {
            \exp_not:N \mode_leave_vertical:
            \exp_not:n { \skip_horizontal:n }
              { \dim_use:c { l__enumext_fake_item_indent_ \__enumext_level: _dim } }
            \ignorespaces
          }
      }
   }
\cs_set_protected:Nn \__enumext_keyans_fake_item:
  {
    \dim_compare:nNnT
      { \l__enumext_fake_item_indent_v_dim } > { \c_zero_dim }
      {
        \tl_set:Ne \l__enumext_fake_item_indent_v_tl
          {
            \exp_not:N \mode_leave_vertical:
            \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_v_dim
          }
      }
   }
\cs_set_protected:Nn \__enumext_fake_item_vii:
  {
    \dim_compare:nNnT
      { \l__enumext_fake_item_indent_vii_dim } > { \c_zero_dim }
      {
        \tl_set:Ne \l__enumext_fake_item_indent_vii_tl
          {
            \exp_not:N \mode_leave_vertical:
            \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_vii_dim
          }
      }
   }
\cs_set_protected:Nn \__enumext_fake_item_viii:
  {
    \dim_compare:nNnT
      { \l__enumext_fake_item_indent_viii_dim } > { \c_zero_dim }
      {
        \tl_set:Ne \l__enumext_fake_item_indent_viii_tl
          {
            \exp_not:N \mode_leave_vertical:
            \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_viii_dim
          }
      }
   }
```

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

11.18 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.

```
\cs_set_protected:Npn \__enumext_tmp:nn #1 #2
  {
    \keys_define:nn { enumext / #1 }
      {
        show-length .bool_set:c = { l__enumext_show_length_#2_bool },
        show-length .initial:n  = false,
      }
  }
\clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }
```

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

11.19 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.

```
\cs_set_protected:Npn \__enumext_tmp:nn #1 #2
  {
    \keys_define:nn { enumext / #1 }
      {
        before  .tl_set:c   = { l__enumext_before_no_starred_key_#2_tl },
        before  .value_required:n = true,
        before* .tl_set:c   = { l__enumext_before_starred_key_#2_tl },
        before* .value_required:n = true,
        after   .tl_set:c   = { l__enumext_after_stop_list_#2_tl },
        after   .value_required:n = true,
        first   .tl_set:c   = { l__enumext_after_list_args_#2_tl },
        first   .value_required:n = true,
      }
  }
\clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }
```

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

11.19.1 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 *second argument* of the list.

```
\cs_new_protected:Nn \__enumext_before_args_exec:
  {
    \tl_use:c { l__enumext_before_starred_key_ \__enumext_level: _tl }
  }
```

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*⟩.

```
\cs_new_protected:Nn \__enumext_before_keys_exec:
  {
    \tl_use:c { l__enumext_before_no_starred_key_ \__enumext_level: _tl }
  }
```

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

```
\cs_new_protected:Nn \__enumext_after_stop_list:
  {
    \tl_use:c { l__enumext_after_stop_list_ \__enumext_level: _tl }
  }
```

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.

```
\cs_new_protected:Nn \__enumext_after_args_exec:
  {
    \tl_use:c { l__enumext_after_list_args_ \__enumext_level: _tl }
  }
```

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

11.19.2 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:

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

```
\cs_new_protected:Nn \__enumext_before_args_exec_v:
  {
    \tl_use:N \l__enumext_before_starred_key_v_tl
  }
```

The function __enumext_before_keys_exec_v: executes the {⟨*code*⟩} set by the before key *"before"* the keyans environment is started in {⟨*arg two*⟩} of the list. The {⟨*code*⟩} is executed *"knowing"* all definition and values provides by ⟨*keys*⟩.

```
\cs_new_protected:Nn \__enumext_before_keys_exec_v:
  {
    \tl_use:N \l__enumext_before_no_starred_key_v_tl
  }
```

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

```
\cs_new_protected:Nn \__enumext_after_stop_list_v:
  {
    \tl_use:N \l__enumext_after_stop_list_v_tl
  }
```

The function __enumext_after_args_exec_v: executes the {⟨*code*⟩} set by the first key after the end of {⟨*arg two*⟩} of the list defining the keyans environment, just before the first occurrence of \item.

```
\cs_new_protected:Nn \__enumext_after_args_exec_v:
  {
    \tl_use:N \l__enumext_after_list_args_v_tl
  }
```

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

11.19.3 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:

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

```
\cs_new_protected:Nn \__enumext_before_args_exec_vii:
  {
    \tl_use:N \l__enumext_before_starred_key_vii_tl
  }
\cs_new_protected:Nn \__enumext_before_args_exec_viii:
  {
    \tl_use:N \l__enumext_before_starred_key_viii_tl
  }
```

The functions __enumext_before_keys_exec_vii: and __enumext_before_keys_exec_viii: executes the {⟨*code*⟩} set by the before key *"before"* in enumext* and keyans* environments is started in {⟨*arg two*⟩} of the list. The {⟨*code*⟩} is executed *"knowing"* all definition and values provides by ⟨*keys*⟩.

```
\cs_new_protected:Nn \__enumext_before_keys_exec_vii:
  {
    \tl_use:N \l__enumext_before_no_starred_key_vii_tl
  }
\cs_new_protected:Nn \__enumext_before_keys_exec_viii:
  {
    \tl_use:N \l__enumext_before_no_starred_key_viii_tl
  }
```

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

```
\cs_new_protected:Nn \__enumext_after_stop_list_vii:
  {
    \tl_use:N \l__enumext_after_stop_list_vii_tl
  }
\cs_new_protected:Nn \__enumext_after_stop_list_viii:
  {
    \tl_use:N \l__enumext_after_stop_list_viii_tl
  }
```

The function `\__enumext_after_args_exec_v:` executes the { ⟨*code*⟩ } set by the `first` key after the end of { ⟨*arg two*⟩ } of the list defining the `keyans` environment, just before the first occurrence of `\item`.

```
\cs_new_protected:Nn \__enumext_after_args_exec_vii:
  {
    \tl_use:N \l__enumext_after_list_args_vii_tl
  }
\cs_new_protected:Nn \__enumext_after_args_exec_viii:
  {
    \tl_use:N \l__enumext_after_list_args_viii_tl
  }
```

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

11.20 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.

```
\cs_set_protected:Npn \__enumext_tmp:nn #1 #2
  {
    \keys_define:nn { enumext / #1 }
      {
        mini-env    .dim_set:c  = { l__enumext_minipage_right_#2_dim },
        mini-env    .value_required:n = true,
        mini-sep    .dim_set:c  = { l__enumext_minipage_hsep_#2_dim },
        mini-sep    .initial:n  = 0.3333em,
        mini-sep    .value_required:n = true,
        columns-sep .dim_set:c  = { l__enumext_columns_sep_#2_dim },
        columns-sep .value_required:n = true,
        columns     .int_set:c  = { l__enumext_columns_#2_int },
        columns     .initial:n  = 1,
        columns     .value_required:n = true,
      }
  }
\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.

```
\cs_set_protected:Npn \__enumext_tmp:nn #1 #2
  {
    \keys_define:nn { enumext / #1 }
      {
        mini-right  .tl_gset:c = { g__enumext_miniright_code_#2_tl },
        mini-right  .value_required:n = true,
        mini-right* .code:n    = {
                                   \bool_gset_true:c { g__enumext_minipage_center_#2_bool }
                                   \keys_set:nn { enumext / #1 } { mini-right = {##1} }
                                 },
        mini-right* .value_required:n = true,
      }
  }
\clist_map_inline:nn { {enumext*}{vii}, {keyans*}{viii} } { \__enumext_tmp:nn #1 }
```

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

11.21 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.

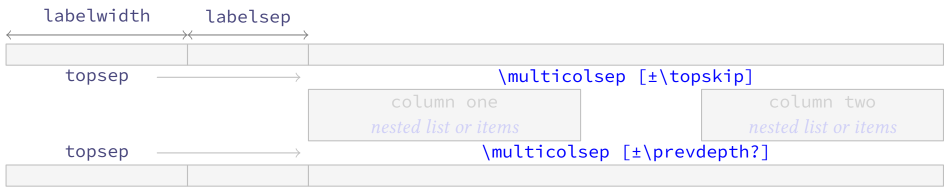


Figure 7: Representation of the vertical space in multicols for a nested level.

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.

11.21.1 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*.

```
\cs_new_protected:Nn \__enumext_multi_set_vskip:
  {
    \skip_set:cn { l__enumext_multicols_above_ \__enumext_level: _skip }
      {
        \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
      }
    \skip_set:cn { l__enumext_multicols_below_ \__enumext_level: _skip }
      {
        \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
      }
    \__enumext_add_pre_parsep:
  }
```

(*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.*

```
\cs_new_protected:Nn \__enumext_add_pre_parsep:
  {
    \int_case:nn { \l__enumext_level_int }
      {
        { 2 }{
               \skip_if_eq:nnF { \l__enumext_parsep_i_skip } { \c_zero_skip }
                 {
                   \skip_add:Nn \l__enumext_multicols_above_ii_skip { \l__enumext_parsep_i_skip }
                 }
             }
        { 3 }{
               \skip_if_eq:nnF { \l__enumext_parsep_ii_skip } { \c_zero_skip }
                 {
                   \skip_add:Nn \l__enumext_multicols_above_iii_skip { \l__enumext_parsep_ii_skip
                 }
             }
        { 4 }{
               \skip_if_eq:nnF { \l__enumext_parsep_iii_skip } { \c_zero_skip }
                 {
                   \skip_add:Nn \l__enumext_multicols_above_iv_skip { \l__enumext_parsep_iii_skip
                 }
             }
      }
  }
```

(*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*⟩.

```
\cs_new_protected:Nn \__enumext_multi_addvspace:
  {
    \__enumext_multi_set_vskip:
    \mode_if_vertical:T
      {
        \skip_add:cn { l__enumext_multicols_above_ \__enumext_level: _skip }
          {
            \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
          }
        \skip_add:cn { l__enumext_multicols_below_ \__enumext_level: _skip }
          {
            \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
          }
      }
    \par\nopagebreak
    \addvspace{ \skip_use:c { l__enumext_multicols_above_ \__enumext_level: _skip } }
  }
```

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

11.21.2 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`.

```
\cs_new_protected:Nn \__enumext_keyans_multi_set_vskip:
  {
    \skip_set:Nn \l__enumext_multicols_above_v_skip
      {
        \l__enumext_topsep_v_skip
      }
    \skip_set:Nn \l__enumext_multicols_below_v_skip
      {
         \l__enumext_topsep_v_skip
      }
  }
\cs_new_protected:Nn \__enumext_keyans_multi_addvspace:
  {
    \__enumext_keyans_multi_set_vskip:
    \mode_if_vertical:T
      {
        \skip_add:Nn \l__enumext_multicols_above_v_skip
          {
            \skip_use:N \l__enumext_partopsep_v_skip
          }
        \skip_add:Nn \l__enumext_multicols_below_v_skip
          {
            \skip_use:N \l__enumext_partopsep_v_skip
          }
      }
    \par\nopagebreak
    \addvspace{ \l__enumext_multicols_above_v_skip }
  }
```

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

11.22 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.
Since we want to keep the *“left”* and *“right”* environments *“aligned on top”*, preserving the `\baselineskip` and keep the desired *“spaces”* (`\topsep + [\partopsep]`) it is necessary to *“adjust”* the *“vertical spaces”* for `minipage` environments.

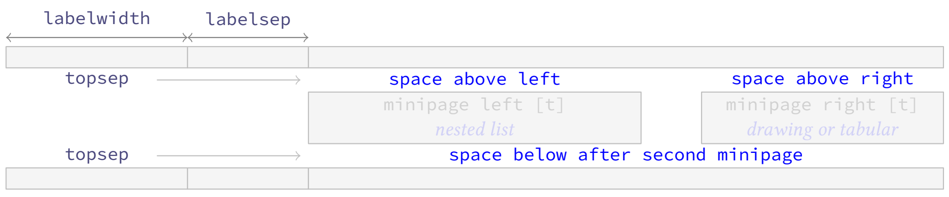


Figure 8: Representation of the minipage spacing adjustment for a nested level.

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

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

11.22.1 Adjustment of vertical spaces for minipage in enumext

__enumext_mini_set_vskip: The function __enumext_mini_set_vskip: will take care of determining the *"adjust"* spaces that we will apply *"above"* and *"below"* the __enumext_mini_env* environment in enumext.

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

First determine if the multicols environment is active by comparing the value of the \l__enumext_columns_X_int variable handled by the columns key, according to this comparison we set the adjusted values for \l__enumext_minipage_left_skip, \l__enumext_minipage_right_skip and \l__enumext_minipage_after_skip.

```
\cs_new_protected:Nn \__enumext_mini_set_vskip:
  {
    \int_compare:nNnTF
      { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } > { 1 }
      {
```

If multicols environment is nested in __enumext_mini_env* environment, we will apply a correction factor to the *vertical spaces* taking into account the value of \topsep of the current level and the value of \parsep of the previous level, if these are zero we will use \strutbox as the basis for the calculations.

```
        \skip_if_eq:nnTF
          { \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip } } { \c_zero_skip }
          {
            \skip_set:Nn \l__enumext_minipage_left_skip
              {
                -0.150\box_dp:N \strutbox
              }
            \skip_set:Nn \l__enumext_minipage_right_skip
              {
                0.695\box_dp:N \strutbox
              }
            \skip_set:Nn \l__enumext_minipage_after_skip
              {
                \box_dp:N \strutbox
              }
            \__enumext_zero_parsep:
          }
          {
            \skip_set:Nn \l__enumext_minipage_left_skip
              {
                \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
              }
            \skip_set:Nn \l__enumext_minipage_right_skip
              {
                0.695\box_dp:N \strutbox
              }
            \skip_set:Nn \l__enumext_minipage_after_skip
              {
                1.85\box_dp:N \strutbox
                + \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
              }
              }
          }
          {
```

If only enumext environment is nested in __enumext_mini_env* environment, we will apply a correction factor to the *vertical spaces* taking into account the value of \topsep, if this is zero we will use \strutbox as the basis for the calculations.

```
            \skip_if_eq:nnTF
              { \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip } } { \c_zero_skip }
              {
                \skip_set:Nn \l__enumext_minipage_left_skip
                  {
                    0.5\box_dp:N \strutbox
                    - \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
                  }
                \skip_set:Nn \l__enumext_minipage_right_skip
                  {
                    \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
                  }
                \skip_set:Nn \l__enumext_minipage_after_skip
                  {
                    1.6\box_dp:N \strutbox
                  }
              }
              {
                \skip_set:Nn \l__enumext_minipage_left_skip
                  {
                    0.5875\box_dp:N \strutbox
                    - \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
                  }
                \skip_set:Nn \l__enumext_minipage_right_skip
                  {
                    + \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
                    + \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
                  }
                \skip_set:Nn \l__enumext_minipage_after_skip
                  {
                    0.325\box_dp:N \strutbox
                    + \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
                  }
              }
          }
    }
```

(*End of definition for* __enumext_mini_set_vskip:.)

__enumext_zero_parsep: The function __enumext_zero_parsep: *"adjusted"* the value of \l__enumext_minipage_after_-skip detecting the value of \parsep from the previous level. This is necessary since \parsep from the previous level affects the *vertical spaces* and this is noticeable when using the nosep or noitemsep keys.

```
\cs_new_protected:Nn \__enumext_zero_parsep:
  {
    \int_case:nn { \l__enumext_level_int }
      {
        { 2 }{
               \skip_if_eq:nnF { \l__enumext_parsep_i_skip } { \c_zero_skip }
                 {
                   \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
                 }
             }
        { 3 }{
               \skip_if_eq:nnF { \l__enumext_parsep_ii_skip } { \c_zero_skip }
                 {
                   \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
                 }
             }
        { 4 }{
               \skip_if_eq:nnF { \l__enumext_parsep_iii_skip } { \c_zero_skip }
                 {
                   \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
                 }
             }
    }
  }
```

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

`\__enumext_mini_addvspace:`

The function `\__enumext_mini_addvspace:` will apply the spaces set using `\addvspace` *"above"* the `__enumext_mini_env*` environment in `enumext`, taking into account whether TEX is in ⟨*horizontal mode*⟩ or ⟨*vertical mode*⟩. For the latter we will make some adjustments since the `\partopsep` parameter comes into play and this affects the *vertical spacing*.

```
\cs_new_protected:Nn \__enumext_mini_addvspace:
  {
    \__enumext_mini_set_vskip:
    \mode_if_vertical:T
      {
        \skip_add:Nn \l__enumext_minipage_left_skip
          {
            \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
          }
        \skip_add:Nn \l__enumext_minipage_after_skip
          {
            \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
          }
      }
    \par\nopagebreak
    \addvspace { \l__enumext_minipage_left_skip }
  }
```

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

11.22.2 Adjustment of vertical spaces for minipage in keyans

`\__enumext_keyans_mini_set_vskip:`

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

```
\cs_new_protected:Nn \__enumext_keyans_mini_set_vskip:
  {
    \skip_zero_new:N \l__enumext_minipage_after_skip
    \skip_zero_new:N \l__enumext_minipage_left_skip
    \skip_zero_new:N \l__enumext_minipage_right_skip
    \int_compare:nNnTF { \l__enumext_columns_v_int } > { 1 }
      {
        \skip_if_eq:nnTF { \l__enumext_topsep_v_skip } { \c_zero_skip }
          {
            \skip_set:Nn \l__enumext_minipage_left_skip { -0.25\box_dp:N \strutbox }
            \skip_set:Nn \l__enumext_minipage_right_skip { 0.705\box_dp:N \strutbox }
            \skip_set:Nn \l__enumext_minipage_after_skip { \box_dp:N \strutbox }
            \skip_if_eq:nnF { \l__enumext_parsep_i_skip } { \c_zero_skip }
              {
                \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
              }
          }
          {
            \skip_set:Nn \l__enumext_minipage_left_skip
              {
                \skip_use:N \l__enumext_topsep_v_skip
              }
            \skip_set:Nn \l__enumext_minipage_right_skip
              {
                0.705\box_dp:N \strutbox
              }
            \skip_set:Nn \l__enumext_minipage_after_skip
              {
                1.85\box_dp:N \strutbox + \l__enumext_topsep_v_skip
              }
          }
      }
      {
        \skip_if_eq:nnTF { \l__enumext_topsep_v_skip } { \c_zero_skip }
          {
            \skip_set:Nn \l__enumext_minipage_left_skip
              {
                0.5\box_dp:N \strutbox
                + \l__enumext_partopsep_v_skip
              }
            \skip_set:Nn \l__enumext_minipage_right_skip
              {
                \l__enumext_partopsep_v_skip
              }
            \skip_set:Nn \l__enumext_minipage_after_skip { 1.6\box_dp:N \strutbox }
          }
          {
            \skip_set:Nn \l__enumext_minipage_left_skip
              {
                0.5875\box_dp:N \strutbox - \l__enumext_partopsep_v_skip
              }
            \skip_set:Nn \l__enumext_minipage_right_skip
              {
                \l__enumext_topsep_v_skip + \l__enumext_partopsep_v_skip
              }
            \skip_set:Nn \l__enumext_minipage_after_skip
              {
                0.325\box_dp:N \strutbox + \l__enumext_topsep_v_skip
              }
          }
      }
  }
```

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

`\__enumext_keyans_mini_addvspace:` The function `\__enumext_keyans_mini_addvspace:` will apply the spaces set using `\addvspace` *"above"* the `__enumext_mini_env*` environment in `keyans`, taking into account whether TEX is in ⟨*horizontal mode*⟩ or ⟨*vertical mode*⟩. For the latter we will make some adjustments since the `\partopsep` parameter comes into play and this affects the *vertical spacing*. The implementation of this function is the same as the one used in `enumext`.

```
\cs_new_protected:Nn \__enumext_keyans_mini_addvspace:
  {
    \__enumext_keyans_mini_set_vskip:
    \mode_if_vertical:T
      {
        \skip_add:Nn \l__enumext_minipage_left_skip
          {
            \l__enumext_partopsep_v_skip
          }
        \skip_add:Nn \l__enumext_minipage_after_skip
          {
            \l__enumext_partopsep_v_skip
          }
      }
    \par\nopagebreak
    \addvspace { \l__enumext_minipage_left_skip }
  }
```

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

11.22.3 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_env*` environment in `enumext*` and `keyans*`.

```
\cs_new_protected:Nn \__enumext_mini_set_vskip_vii:
  {
    \skip_zero_new:N \l__enumext_minipage_left_skip
    \skip_gzero_new:N \g__enumext_minipage_right_skip
    \skip_gzero_new:N \g__enumext_minipage_after_skip
    \skip_if_eq:nnTF { \l__enumext_topsep_vii_skip } { \c_zero_skip }
      {
        \skip_set:Nn \l__enumext_minipage_left_skip { 0.5\box_dp:N \strutbox }
        \skip_gset:Nn \g__enumext_minipage_right_skip { 0.325\box_dp:N \strutbox }
      }
      {
        \skip_set:Nn \l__enumext_minipage_left_skip { 0.5875\box_dp:N \strutbox }
        \skip_gset:Nn \g__enumext_minipage_right_skip
```

```
          {
            \l__enumext_topsep_vii_skip
          }
        \skip_gset:Nn \g__enumext_minipage_after_skip
          {
            0.325\box_dp:N \strutbox + \l__enumext_topsep_vii_skip
          }
      }
  }
\cs_new_protected:Nn \__enumext_mini_set_vskip_viii:
  {
    \skip_zero_new:N \l__enumext_minipage_after_skip
    \skip_zero_new:N \l__enumext_minipage_left_skip
    \skip_zero_new:N \l__enumext_minipage_right_skip
    \skip_if_eq:nnTF { \l__enumext_topsep_viii_skip } { \c_zero_skip }
      {
        \skip_set:Nn \l__enumext_minipage_left_skip
          {
            0.5\box_dp:N \strutbox
          }
        \skip_set:Nn \l__enumext_minipage_right_skip
          {
            \l__enumext_partopsep_viii_skip
          }
        \skip_set:Nn \l__enumext_minipage_after_skip
          {
            1.6\box_dp:N \strutbox
          }
      }
      {
        \skip_set:Nn \l__enumext_minipage_left_skip
          {
            0.5875\box_dp:N \strutbox
          }
        \skip_set:Nn \l__enumext_minipage_right_skip
          {
            \l__enumext_topsep_viii_skip
          }
        \skip_set:Nn \l__enumext_minipage_after_skip
          {
            0.325\box_dp:N \strutbox + \l__enumext_topsep_viii_skip
          }
       }
  }
```

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

`\__enumext_mini_addvspace_vii:`
`\__enumext_mini_addvspace_viii:`

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

```
\cs_new_protected:Nn \__enumext_mini_addvspace_vii:
  {
    \__enumext_mini_set_vskip_vii:
    \par\nopagebreak
    \addvspace { \l__enumext_minipage_left_skip }
  }
\cs_new_protected:Nn \__enumext_mini_addvspace_viii:
  {
    \__enumext_mini_set_vskip_viii:
    \par\nopagebreak
    \addvspace { \l__enumext_minipage_left_skip }
  }
```

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

11.22.4 The command `\miniright`

The command `\miniright` will close the `__enumext_mini_env*` environment on the *"left side"*, open the `__enumext_mini_env*` environment on the *"right side"* adding the *adjusted vertical space*. By default we will add `\centering` when starting the *"right side"* environment. The *starred argument* '`*`' inhibits the use of \centering command i.e. the usual LaTeX justification is maintained in the __enumext_mini_env* on the *"right side"*.

\miniright First we will perform some checks to prevent the command from being executed outside the enumext environment or from being executed inside the keyanspic environment, then we call the internal functions for the enumext and keyans environments.

```
\NewDocumentCommand \miniright { s }
  {
    \int_compare:nNnT { \l__enumext_keyans_pic_level_int } = { 1 }
      {
        \msg_error:nnn { enumext } { wrong-miniright-place }
      }
    \int_compare:nNnT { \l__enumext_level_int } = { 0 }
      {
        \msg_error:nnn { enumext } { wrong-miniright-place }
      }
    \int_compare:nNnTF { \l__enumext_keyans_level_int } = { 1 }
      {
        \__enumext_keyans_mini_right_cmd:n {#1}
      }
      { \__enumext_mini_right_cmd:n {#1} }
  }
```

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

__enumext_mini_right_cmd:n The function __enumext_mini_right_cmd:n takes as argument the *starred* '*' of the \miniright command in the enumext environment. We check if the mini-env key is active via the variable \l__enumext_minipage_right_X_dim, if so we close the multicols environment with the __enumext_mini_env* environment on the *"left side"*, then we open the __enumext_mini_env* environment on the *"right side"*, apply our adjusted *"vertical spaces"*, followed by adding the \centering command when the starred argument '*' is not present and set zero \g__enumext_minipage_stat_int, otherwise we return an error.

```
\cs_new_protected:Npn \__enumext_mini_right_cmd:n #1
  {
    \dim_compare:nNnTF
      { \dim_use:c { l__enumext_minipage_right_ \__enumext_level: _dim } } > { \c_zero_dim }
      {
        \__enumext_multicols_stop:
        \end{__enumext_mini_env*}
        \hfill
        \begin{__enumext_mini_env*}
          { \dim_use:c { l__enumext_minipage_right_ \__enumext_level: _dim } }
          \par\addvspace { \l__enumext_minipage_right_skip }
          \bool_if:nF {#1}
            {
              \centering
            }
          \int_gzero:N \g__enumext_minipage_stat_int
      }
      { \msg_error:nnn { enumext } { wrong-miniright-use } }
  }
```

(*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.

```
\cs_new_protected:Npn \__enumext_keyans_mini_right_cmd:n #1
  {
    \dim_compare:nNnTF { \l__enumext_minipage_right_v_dim } > { \c_zero_dim }
      {
        \__enumext_keyans_multicols_stop:
        \end{__enumext_mini_env*}
        \hfill
        \begin{__enumext_mini_env*}{ \l__enumext_minipage_right_v_dim }
          \par\addvspace { \l__enumext_minipage_right_skip }
          \bool_if:nF {#1}
            {
              \centering
            }
          \int_gzero:N \g__enumext_minipage_stat_int
        }
        { \msg_error:nnn { enumext } { wrong-miniright-use } }
    }
```

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

11.23 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.

```
\cs_set_protected:Npn \__enumext_tmp:nn #1 #2
  {
    \keys_define:nn { enumext / #1 }
      {
        above  .skip_set:c = { l__enumext_vspace_above_#2_skip },
        above  .value_required:n = true,
        above* .code:n     = \bool_set_true:c { l__enumext_vspace_a_star_#2_bool }
                             \keys_set:nn { enumext / #1 } { above = {##1} },
        above* .value_required:n = true,
        below  .skip_set:c = { l__enumext_vspace_below_#2_skip },
        below  .value_required:n = true,
        below* .code:n     = \bool_set_true:c { l__enumext_vspace_b_star_#2_bool }
                             \keys_set:nn { enumext / #1 } { below = {##1} },
        below* .value_required:n = true,
      }
  }
\clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }
```

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

11.23.1 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.

```
\cs_new_protected:Nn \__enumext_vspace_above:
  {
    \skip_if_eq:nnF
      { \skip_use:c { l__enumext_vspace_above_ \__enumext_level: _skip } } { \c_zero_skip }
      {
        \bool_if:cTF { l__enumext_vspace_a_star_ \__enumext_level: _bool }
          {
            \vspace*{ \skip_use:c { l__enumext_vspace_above_ \__enumext_level: _skip } }
          }
          {
            \vspace { \skip_use:c { l__enumext_vspace_above_ \__enumext_level: _skip } }
          }
      }
  }
```

(*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.

```
\cs_new_protected:Nn \__enumext_vspace_below:
  {
    \skip_if_eq:nnF
      { \skip_use:c { l__enumext_vspace_below_ \__enumext_level: _skip } } { \c_zero_skip }
      {
        \bool_if:cTF { l__enumext_vspace_b_star_ \__enumext_level: _bool }
          {
            \vspace*{ \skip_use:c { l__enumext_vspace_below_ \__enumext_level: _skip } }
          }
          {
            \vspace { \skip_use:c { l__enumext_vspace_below_ \__enumext_level: _skip } }
          }
      }
  }
```

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

11.23.2 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.

```
\cs_new_protected:Nn \__enumext_vspace_above_v:
  {
    \skip_if_eq:nnF { \l__enumext_vspace_above_v_skip } { \c_zero_skip }
      {
        \bool_if:NTF \l__enumext_vspace_a_star_v_bool
          {
            \vspace*{ \l__enumext_vspace_above_v_skip }
          }
          { \vspace { \l__enumext_vspace_above_v_skip } }
      }
  }
```

(*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.

```
\cs_new_protected:Nn \__enumext_vspace_below_v:
  {
    \skip_if_eq:nnF { \l__enumext_vspace_below_v_skip } { \c_zero_skip }
      {
        \bool_if:NTF \l__enumext_vspace_b_star_v_bool
          {
            \vspace*{ \l__enumext_vspace_below_v_skip }
          }
          { \vspace { \l__enumext_vspace_below_v_skip } }
      }
  }
```

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

11.23.3 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.

```
\cs_new_protected:Nn \__enumext_vspace_above_vii:
  {
    \skip_if_eq:nnF { \l__enumext_vspace_above_vii_skip } { \c_zero_skip }
      {
        \bool_if:NTF \l__enumext_vspace_a_star_vii_bool
          {
            \vspace*{ \l__enumext_vspace_above_vii_skip }
          }
          { \vspace { \l__enumext_vspace_above_vii_skip } }
      }
  }
\cs_new_protected:Nn \__enumext_vspace_above_viii:
  {
    \skip_if_eq:nnF { \l__enumext_vspace_above_viii_skip } { \c_zero_skip }
      {
        \bool_if:NTF \l__enumext_vspace_a_star_viii_bool
          {
            \vspace*{ \l__enumext_vspace_above_viii_skip }
          }
          { \vspace { \l__enumext_vspace_above_viii_skip } }
      }
  }
```

(*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.

```
\cs_new_protected:Nn \__enumext_vspace_below_vii:
  {
    \skip_if_eq:nnF { \l__enumext_vspace_below_vii_skip } { \c_zero_skip }
      {
        \bool_if:NTF \l__enumext_vspace_b_star_vii_bool
          {
```

```
          \vspace*{ \l__enumext_vspace_below_vii_skip }
        }
        { \vspace { \l__enumext_vspace_below_vii_skip } }
    }
  }
\cs_new_protected:Nn \__enumext_vspace_below_viii:
  {
    \skip_if_eq:nnF { \l__enumext_vspace_below_viii_skip } { \c_zero_skip }
      {
        \bool_if:NTF \l__enumext_vspace_b_star_viii_bool
          {
            \vspace*{ \l__enumext_vspace_below_viii_skip }
          }
          { \vspace { \l__enumext_vspace_below_viii_skip } }
      }
  }
```

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

11.24 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*.

```
\cs_set_protected:Npn \__enumext_tmp:n #1
  {
    \keys_define:nn { enumext / #1 }
      {
        series  .str_set:N = \l__enumext_series_str,
        series  .value_required:n = true,
        resume  .code:n = \__enumext_resume_series:n {##1},
        resume* .code:n = \__enumext_resume_starred:,
        resume* .value_forbidden:n = true,
      }
  }
\clist_map_inline:nn { level-1, enumext* } { \__enumext_tmp:n {#1} }
```

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

11.24.1 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.

```
\cs_new:Npn \__enumext_filter_series:n #1
  {
    \use:e
      {
        \keyval_parse:NNn
          \__enumext_filter_series_key:n
          \__enumext_filter_series_pair:nn {#1}
      }
  }
```

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.

```
\cs_new:Npn \__enumext_filter_series_key:n #1
  {
    \str_case:nnF {#1}
      {
        { resume   } {}
        { resume*  } {}
        { base-fix } {}
      }
      { , { \exp_not:n {#1} } }
  }
```

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

```
\cs_new:Npn \__enumext_filter_series_pair:nn #1#2
  {
    \str_case:nnF {#1}
      {
        { series } {}
        { resume } {}
        { start  } {}
        { save-ans } {}
        { save-key } {}
      }
      { , { \exp_not:n {#1} } = { \exp_not:n {#2} } }
  }
```

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

`\__enumext_parse_series:n`
`\__enumext_resume_last:n`

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

The value of boolean variable `\l__enumext_resume_active_bool` is set to true by the function `\__enumext_resume_counter:n` which is used by the keys `resume` and `resume*`, in this case we must Make sure it is set to false so that it does not overwrite the default filtered ⟨*keys*⟩. This function is passed to the function `\__enumext_parse_keys:n` in the `enumext` environment definition (§11.38) and to the function `\__enumext_parse_keys_vii:n` in the `enumext*` environment definition (§11.42).

```
\cs_new_protected:Npn \__enumext_parse_series:n #1
  {
    \str_if_empty:NTF \l__enumext_series_str
      {
        \bool_if:NF \l__enumext_resume_active_bool
          {
            \__enumext_resume_last:n {#1}
          }
      }
      {
        \tl_gclear_new:c { g__enumext_series_ \l__enumext_series_str _tl }
        \tl_gset:ce { g__enumext_series_ \l__enumext_series_str _tl }
          { \__enumext_filter_series:n {#1} }
        \int_if_exist:cF { g__enumext_series_ \l__enumext_series_str _int }
          {
            \int_new:c { g__enumext_series_ \l__enumext_series_str _int }
          }
      }
  }
```

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.

```
\cs_new_protected:Npn \__enumext_resume_last:n #1
  {
    \bool_if:NT \l__enumext_standar_first_bool
      {
        \tl_gclear:N \g__enumext_standar_series_tl
        \tl_gset:Ne \g__enumext_standar_series_tl { \__enumext_filter_series:n {#1} }
      }
    \bool_if:NT \l__enumext_starred_first_bool
      {
        \tl_gclear:N \g__enumext_starred_series_tl
        \tl_gset:Ne \g__enumext_starred_series_tl { \__enumext_filter_series:n {#1} }
      }
  }
```

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

11.24.2 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 (§11.38) and the enumext* environment definition (§11.42).

```
\cs_new_protected:Nn \__enumext_resume_save_counter:
  {
    \bool_if:NT \g__enumext_standar_bool
      {
        \tl_if_empty:NF \l__enumext_series_str
          {
            \int_gset_eq:cN
              { g__enumext_series_ \l__enumext_series_str _int } \value{enumXi}
          }
        \tl_if_empty:NTF \l__enumext_resume_name_tl
          {
            \str_if_empty:NT \l__enumext_series_str
              {
                \int_gset_eq:NN \g__enumext_resume_int \value{enumXi}
              }
          }
          {
            \int_if_exist:cT { g__enumext_series_ \l__enumext_resume_name_tl _int }
              {
                \int_gset_eq:cN
                  { g__enumext_series_ \l__enumext_resume_name_tl _int } \value{enumXi}
              }
          }
        \int_if_exist:cT { g__enumext_resume_ \l__enumext_store_name_tl _int }
          {
            \int_gset_eq:cN
              { g__enumext_resume_ \l__enumext_store_name_tl _int } \value{enumXi}
          }
      }
    \bool_if:NT \g__enumext_starred_bool
      {
        \tl_if_empty:NF \l__enumext_series_str
          {
            \int_gset_eq:cN
              { g__enumext_series_ \l__enumext_series_str _int } \value{enumXvii}
          }
        \tl_if_empty:NTF \l__enumext_resume_name_tl
          {
            \str_if_empty:NT \l__enumext_series_str
              {
                \int_gset_eq:NN \g__enumext_resume_vii_int \value{enumXvii}
              }
          }
          {
            \int_if_exist:cT { g__enumext_series_ \l__enumext_resume_name_tl _int }
              {
                \int_gset_eq:cN
                  { g__enumext_series_ \l__enumext_resume_name_tl _int } \value{enumXvii}
              }
          }
        \int_if_exist:cT { g__enumext_resume_ \l__enumext_store_name_tl _int }
          {
            \int_gset_eq:cN
              { g__enumext_resume_ \l__enumext_store_name_tl _int } \value{enumXvii}
          }
      }
  }
```

(*End of definition for* __enumext_resume_save_counter:.)

11.24.3 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.

```
\cs_new_protected:Npn \__enumext_resume_series:n #1
  {
    \tl_if_empty:nTF {#1}
      {
        \__enumext_resume_counter:n { }
      }
      {
        \tl_if_exist:cTF { g__enumext_series_ \tl_to_str:n {#1} _tl }
          {
            \__enumext_resume_counter:n {#1}
            \bool_if:NT \g__enumext_standar_bool
              {
                \keys_set:nv { enumext / level-1 }
                  { g__enumext_series_ \tl_to_str:n {#1} _tl }
              }
            \bool_if:NT \g__enumext_starred_bool
              {
                \keys_set:nv { enumext / enumext* }
                  { g__enumext_series_ \tl_to_str:n {#1} _tl }
              }
          }
          {
            \bool_if:NT \g__enumext_standar_bool
              {
                \msg_error:nnn { enumext } { unknown-series } {#1}
              }
            \bool_if:NT \g__enumext_starred_bool
              {
                \msg_error:nnn { enumext } { unknown-series } {#1}
              }
          }
      }
  }
```

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

__enumext_resume_counter:n
__enumext_resume_counter:
__enumext_resume_counter_series:
__enumext_resume_counter_save_ans:

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

```
\cs_new_protected:Npn \__enumext_resume_counter:n #1
  {
    \bool_set_true:N \l__enumext_resume_active_bool
    \tl_set:Nn \l__enumext_resume_name_tl {#1}
    \tl_if_empty:NTF \l__enumext_resume_name_tl
      {
        \__enumext_resume_counter:
      }
      {
        \__enumext_resume_counter_series:
      }
    \__enumext_resume_counter_save_ans:
  }
```

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.

```
\cs_new_protected:Nn \__enumext_resume_counter:
  {
    \bool_if:NT \g__enumext_standar_bool
      {
        \int_gincr:N \g__enumext_resume_int
        \int_set_eq:NN \l__enumext_start_i_int \g__enumext_resume_int
      }
    \bool_if:NT \g__enumext_starred_bool
      {
        \int_gincr:N \g__enumext_resume_vii_int
        \int_set_eq:NN \l__enumext_start_vii_int \g__enumext_resume_vii_int
      }
  }
```

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.

```
\cs_new_protected:Nn \__enumext_resume_counter_series:
  {
    \bool_if:NT \g__enumext_standar_bool
      {
        \int_set:Nn \l__enumext_start_i_int
          {
            \int_use:c { g__enumext_series_ \l__enumext_resume_name_tl _int } + 1
          }
      }
    \bool_if:NT \g__enumext_starred_bool
      {
        \int_set:Nn \l__enumext_start_vii_int
          {
            \int_use:c { g__enumext_series_ \l__enumext_resume_name_tl _int } + 1
          }
      }
  }
```

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.

```
\cs_new_protected:Nn \__enumext_resume_counter_save_ans:
  {
    \bool_lazy_and:nnT
      { \bool_if_p:N \l__enumext_standar_first_bool }
      { \bool_if_p:N \l__enumext_store_active_bool }
      {
        \int_set:Nn \l__enumext_start_i_int
          {
            \int_use:c { g__enumext_resume_ \l__enumext_store_name_tl _int } + 1
          }
      }
    \bool_lazy_and:nnT
      { \bool_if_p:N \l__enumext_starred_first_bool }
      { \bool_if_p:N \l__enumext_store_active_bool }
      {
        \int_set:Nn \l__enumext_start_vii_int
          {
            \int_use:c { g__enumext_resume_ \l__enumext_store_name_tl _int } + 1
          }
      }
  }
```

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

11.24.4 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.

```
\cs_new_protected:Nn \__enumext_resume_starred:
  {
    \bool_if:NT \g__enumext_standar_bool
      {
        \tl_if_empty:NF \g__enumext_standar_series_tl
          {
            \__enumext_resume_counter:n { }
            \keys_set:nV { enumext / level-1 } \g__enumext_standar_series_tl
          }
      }
    \bool_if:NT \g__enumext_starred_bool
      {
        \tl_if_empty:NF \g__enumext_starred_series_tl
          {
            \__enumext_resume_counter:n { }
            \keys_set:nV { enumext / enumext* } \g__enumext_starred_series_tl
          }
      }
  }
```

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

11.25 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.

11.25.1 Setting save-ans key

save-ans

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

```
\cs_set_protected:Npn \__enumext_tmp:n #1
  {
    \keys_define:nn { enumext / #1 }
      {
        save-ans .code:n = \__enumext_storing_set:n {##1},
        save-ans .value_required:n = true,
      }
  }
\clist_map_inline:nn { level-1, enumext* } { \__enumext_tmp:n {#1} }
```

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

11.25.2 Internal functions for save-ans key

__enumext_start_save_ans_msg:
__enumext_stop_save_ans_msg:

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

```
\cs_new_protected:Nn \__enumext_start_save_ans_msg:
  {
    \msg_term:nnVV { enumext } { save-ans-log }
      \g__enumext_envir_name_tl \l__enumext_store_name_tl
  }
\cs_new_protected:Nn \__enumext_stop_save_ans_msg:
  {
    \msg_term:nnVV { enumext } { save-ans-log-hook }
      \g__enumext_envir_name_tl \g__enumext_store_name_tl
  }
```

(*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.

```
\cs_new_protected:Npn \__enumext_storing_set:n #1
  {
    \tl_set:Ne \l__enumext_store_name_tl {#1}
    \tl_if_empty:NTF \l__enumext_store_name_tl
      {
        \bool_lazy_or:nnT
          { \l__enumext_standar_first_bool } { \l__enumext_starred_first_bool }
          {
            \msg_error:nnV { enumext } { save-ans-empty } \g__enumext_envir_name_tl
          }
      }
      {
        \bool_lazy_or:nnT
          { \l__enumext_standar_first_bool } { \l__enumext_starred_first_bool }
          {
            \__enumext_start_save_ans_msg:
            \__enumext_storing_exec:
          }
      }
  }
```

The function `\__enumext_storing_exec:` will set to true the variable `\l__enumext_store_active_bool` which activates the use of the `\anskey` command and the `keyans`, `keyans*` and `keyanspic` environments and will set to true the variable `\l__enumext_check_answers_bool` used for checking answers by the `check-ans` and `no-store` keys, copy `{`⟨*store name*⟩`}` into the global variable `\g__enumext_store_name_tl` and execute the function `\__enumext_anskey_env_make:V` creating the environment `anskey*` (§11.30). 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*`.

```
\cs_new_protected:Nn \__enumext_storing_exec:
  {
    \bool_set_true:N \l__enumext_store_active_bool
    \bool_set_true:N \l__enumext_check_answers_bool
    \tl_gset:NV \g__enumext_store_name_tl \l__enumext_store_name_tl
    \__enumext_anskey_env_make:V \l__enumext_store_name_tl
    \prop_if_exist:cF { g__enumext_ \l__enumext_store_name_tl _prop }
      {
        \msg_log:nnV { enumext } { store-prop } \l__enumext_store_name_tl
        \prop_new:c { g__enumext_ \l__enumext_store_name_tl _prop }
      }
    \seq_if_exist:cF { g__enumext_ \l__enumext_store_name_tl _seq }
      {
        \msg_log:nnV { enumext } { store-seq } \l__enumext_store_name_tl
        \seq_new:c { g__enumext_ \l__enumext_store_name_tl _seq }
      }
    \int_if_exist:cF { g__enumext_resume_ \l__enumext_store_name_tl _int }
      {
        \msg_log:nnV { enumext } { store-int } \l__enumext_store_name_tl
        \int_new:c { g__enumext_resume_ \l__enumext_store_name_tl _int }
      }
  }
```

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

11.25.3 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.

11.25.4 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.

```
\cs_set_protected:Npn \__enumext_tmp:n #1
  {
    \keys_define:nn { enumext / #1 }
      {
        check-ans .bool_set:N = \l__enumext_check_ans_key_bool,
        check-ans .initial:n  = false,
        check-ans .value_required:n = true,
        no-store  .code:n = {
                              \bool_set_false:N \l__enumext_check_answers_bool
                              \bool_set_false:N \l__enumext_check_ans_key_bool
                            },
        no-store  .value_forbidden:n = true,
      }
  }
\clist_map_inline:nn
  {
    level-1, level-2, level-3, level-4, enumext*
  }
  { \__enumext_tmp:n {#1} }
```

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

11.25.5 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.

```
\cs_new_protected:Nn \__enumext_check_ans_active:
  {
    \tl_if_empty:NF \l__enumext_store_name_tl
      {
        \bool_if:NT \l__enumext_check_answers_bool
          {
            \__enumext_check_ans_level:
          }
      }
  }
```

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"* .

```
\cs_new_protected:Nn \__enumext_check_ans_level:
  {
    \int_case:nn { \l__enumext_level_int }
      {
        { 1 }{
                \bool_lazy_all:nT
                  {
                    { \bool_if_p:N \g__enumext_starred_bool }
                    { \int_compare_p:nNn { \l__enumext_level_h_int } = { 1 } }
                  }
                  {
                    \int_gdecr:N \g__enumext_item_number_int
                    \bool_set_false:N \l__enumext_item_number_bool
                  }
              }
        { 2 }{
                \int_gdecr:N \g__enumext_item_number_int
                \bool_set_false:N \l__enumext_item_number_bool
              }
        { 3 }{
                \int_gdecr:N \g__enumext_item_number_int
                \bool_set_false:N \l__enumext_item_number_bool
              }
        { 4 }{
                \int_gdecr:N \g__enumext_item_number_int
                \bool_set_false:N \l__enumext_item_number_bool
                }
        }
```

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.

```
      \int_case:nn { \l__enumext_level_h_int }
        {
          { 1 }{
                \bool_lazy_all:nT
                  {
                    { \bool_if_p:N \g__enumext_standar_bool }
                    { \int_compare_p:nNn { \l__enumext_level_int } = { 1 } }
                  }
                  {
                    \int_gdecr:N \g__enumext_item_number_int
                    \bool_set_false:N \l__enumext_item_number_bool
                  }
                }
        }
    }
```

(*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.

```
  \cs_new_protected:Nn \__enumext_check_ans_key_hook:
    {
      \bool_lazy_and:nnT
        { \bool_if_p:N \l__enumext_check_ans_key_bool }
        { \bool_if_p:N \g__enumext_standar_bool }
        {
          \bool_gset_true:N \g__enumext_check_ans_key_bool
        }
      \bool_lazy_and:nnT
        { \bool_if_p:N \l__enumext_check_ans_key_bool }
        { \bool_if_p:N \g__enumext_starred_bool }
        {
          \bool_gset_true:N \g__enumext_check_ans_key_bool
        }
    }
```

(*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:`.

```
  \cs_new_protected:Nn \__enumext_item_answer_diff:
    {
      \int_gset:Nn \g__enumext_item_answer_diff_int
        {
          \int_sign:n { \g__enumext_item_number_int - \g__enumext_item_anskey_int }
        }
    }
```

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

__enumext_check_ans_show:
__enumext_check_ans_msg_less:
__enumext_check_ans_msg_same_ok:
__enumext_check_ans_msg_greater:

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

```
  \cs_new_protected:Nn \__enumext_check_ans_show:
    {
      \int_case:nn { \g__enumext_item_answer_diff_int }
        {
          { -1 }{ \__enumext_check_ans_msg_less:    }
          {  0 }{ \__enumext_check_ans_msg_same_ok: }
          {  1 }{ \__enumext_check_ans_msg_greater: }
        }
    }
```

```
\cs_new_protected:Nn \__enumext_check_ans_msg_less:
  {
    \msg_warning:nneee { enumext } { item-less-answer } { \g__enumext_store_name_tl }
      { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
  }
\cs_new_protected:Nn \__enumext_check_ans_msg_same_ok:
  {
    \msg_term:nneee { enumext } { items-same-answer } { \g__enumext_store_name_tl }
      { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
  }
\cs_new_protected:Nn \__enumext_check_ans_msg_greater:
  {
    \msg_warning:nneee { enumext } { item-greater-answer } { \g__enumext_store_name_tl }
      { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
  }
```

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

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

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

```
\cs_new_protected:Nn \__enumext_check_ans_log:
  {
    \int_case:nn { \g__enumext_item_answer_diff_int }
      {
        { -1 }{ \__enumext_check_ans_log_msg_less:    }
        {  0 }{ \__enumext_check_ans_log_msg_same_ok: }
        {  1 }{ \__enumext_check_ans_log_msg_greater: }
      }
  }
\cs_new_protected:Nn \__enumext_check_ans_log_msg_less:
  {
    \msg_log:nneee { enumext } { item-less-answer } { \g__enumext_store_name_tl }
      { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
  }
\cs_new_protected:Nn \__enumext_check_ans_log_msg_same_ok:
  {
    \msg_log:nneee { enumext } { items-same-answer }  { \g__enumext_store_name_tl }
      { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
  }
\cs_new_protected:Nn \__enumext_check_ans_log_msg_greater:
  {
    \msg_log:nneee { enumext } { item-greater-answer } { \g__enumext_store_name_tl }
      { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
  }
```

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

11.25.6 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.

```
\cs_new_protected:Npn \__enumext_check_starred_cmd:n #1
  {
    \int_compare:nNnT
      { \g__enumext_check_starred_cmd_int } = { 0 }
      {
        \msg_warning:nnnV
          { enumext } { missing-starred }{ #1 } \l__enumext_check_start_line_env_tl
      }
    \int_compare:nNnT
      { \g__enumext_check_starred_cmd_int } > { 1 }
      {
        \msg_warning:nnnV
          { enumext } { many-starred }{ #1 } \l__enumext_check_start_line_env_tl
      }
    \int_gzero:N \g__enumext_check_starred_cmd_int
    \tl_clear:N \l__enumext_check_start_line_env_tl
  }
```

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

11.26 Keys and functions associated with storage

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

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

```
\cs_set_protected:Npn \__enumext_tmp:n #1
  {
    \keys_define:nn { enumext / #1 }
      {
        wrap-ans   .cs_set_protected:Np = \__enumext_anskey_wrapper:n ##1,
        wrap-ans   .initial:n = \fbox{##1},
        wrap-ans   .value_required:n = true,
        wrap-opt   .cs_set_protected:Np = \__enumext_keyans_wrapper_opt:n ##1,
        wrap-opt   .initial:n = [{##1}],
        wrap-opt   .value_required:n = true,
        save-sep   .tl_set:N  = \l__enumext_store_keyans_item_opt_sep_tl,
        save-sep   .initial:n = {, ~ },
        save-sep   .value_required:n = true,
        mark-ans   .tl_set:N  = \l__enumext_mark_answer_sym_tl,
        mark-ans   .initial:n = \textasteriskcentered,
        mark-ans   .value_required:n = true,
        mark-pos   .choice:,
        mark-pos / left    .code:n = \str_set:Nn \l__enumext_mark_position_str { l },
        mark-pos / right   .code:n = \str_set:Nn \l__enumext_mark_position_str { r },
        mark-pos / unknown .code:n =
                           \msg_error:nneee { enumext } { unknown-choice }
                             { mark-pos } { left, ~ right } { \exp_not:n {##1} },
        mark-pos   .initial:n = right,
        mark-pos   .value_required:n = true,
        show-ans   .bool_set:N = \l__enumext_show_answer_bool,
        show-ans   .initial:n  = false,
        show-ans   .value_required:n = true,
        show-pos   .bool_set:N = \l__enumext_show_position_bool,
        show-pos   .initial:n  = false,
        show-pos   .value_required:n = true,
        mark-ref   .tl_set:N   = \l__enumext_mark_ref_sym_tl,
        mark-ref   .initial:n  = \textasteriskcentered,
        mark-ref   .value_required:n = true,
        save-ref   .bool_set:N = \l__enumext_store_ref_key_bool,
        save-ref   .initial:n  = false,
        save-ref   .value_required:n = true,
      }
  }
\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.

```
\cs_set_protected:Npn \__enumext_tmp:n #1
  {
    \keys_define:nn { enumext / #1 }
      {
        mark-pos .choice:,
        mark-pos / left  .code:n = \str_set:Nn \l__enumext_mark_position_str { l },
        mark-pos / right .code:n = \str_set:Nn \l__enumext_mark_position_str { r },
        mark-pos .initial:n = right,
        mark-pos .value_required:n  = true,
        show-ans .bool_set:N = \l__enumext_show_answer_bool,
        show-ans .initial:n  = false,
        show-ans .value_required:n = true,
        show-pos .bool_set:N = \l__enumext_show_position_bool,
        show-pos .initial:n  = false,
        show-pos .value_required:n = true,
      }
  }
\clist_map_inline:nn { keyans, keyans* } { \__enumext_tmp:n {#1} }
```

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

11.26.1 Store optional arguments of the environments

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

__enumext_store_active_keys:n
__enumext_store_active_keys_vii:n

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

```
\cs_new_protected:Npn \__enumext_store_active_keys:n #1
  {
    \bool_if:cF { l__enumext_store_save_key_ \__enumext_level: _bool }
      {
        \tl_clear:c { l__enumext_save_key_ \__enumext_level: _tl }
        \tl_set:ce
          { l__enumext_store_save_key_ \__enumext_level: _tl }
          { \__enumext_filter_save_key:n {#1} }
      }
  }
\cs_new_protected:Npn \__enumext_store_active_keys_vii:n #1
  {
    \bool_if:NF \l__enumext_store_save_key_vii_bool
      {
        \tl_clear:N \l__enumext_store_save_key_vii_tl
        \tl_set:Ne \l__enumext_store_save_key_vii_tl { \__enumext_filter_save_key:n {#1} }
      }
  }
```

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

11.26.2 Setting save-key key

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

save-key

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

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

```
\cs_set_protected:Npn \__enumext_tmp:n #1
  {
    \keys_define:nn { enumext / enumext* }
      {
        save-key .code:n = \__enumext_parse_save_key_vii:n {##1},
        save-key .value_required:n = true,
      }
    \keys_define:nn { enumext / #1 }
      {
        save-key .code:n = \__enumext_parse_save_key:n {##1},
        save-key .value_required:n = true,
      }
  }
\clist_map_inline:nn { level-1, level-2, level-3, level-4 } { \__enumext_tmp:n {#1} }
```

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

__enumext_parse_save_key:n
__enumext_parse_save_key_vii:n

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

```
\cs_new_protected:Npn \__enumext_parse_save_key:n #1
  {
    \bool_set_true:c { l__enumext_store_save_key_ \__enumext_level: _bool }
    \tl_clear:c { l__enumext_save_key_ \__enumext_level: _tl }
    \tl_set:ce
      { l__enumext_store_save_key_ \__enumext_level: _tl }
      { \__enumext_filter_save_key:n {#1} }
  }
```

```
\cs_new_protected:Npn \__enumext_parse_save_key_vii:n #1
  {
    \bool_set_true:N \l__enumext_store_save_key_vii_bool
    \tl_clear:N \l__enumext_store_save_key_vii_tl
    \tl_set:Ne \l__enumext_store_save_key_vii_tl { \__enumext_filter_save_key:n {#1} }
  }
```

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

11.26.3 Internal functions to store optional arguments

`\__enumext_filter_save_key:n`
`\__enumext_filter_save_key_key:n`
`\__enumext_filter_save_key_pair:nn`

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

```
\cs_new:Npn \__enumext_filter_save_key:n #1
  {
    \use:e
      {
        \keyval_parse:NNn
          \__enumext_filter_save_key_key:n
          \__enumext_filter_save_key_pair:nn {#1}
      }
  }
```

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.

```
\cs_new:Npn \__enumext_filter_save_key_key:n #1
  {
    \str_case:nnF {#1}
      {
        { resume } {} { resume* } {} { no-store } {} { base-fix } {}
      }
      { , { \exp_not:n {#1} } }
  }
```

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.

```
\cs_new:Npn \__enumext_filter_save_key_pair:nn #1#2
  {
    \str_case:nnF {#1}
      {
        { series   } {} { resume    } {} { save-ans } {} { save-ref   } {}
        { save-key } {} { check-ans } {} { show-ans } {} { show-pos   } {}
        { wrap-ans } {} { mark-ans  } {} { wrap-opt } {} { save-sep   } {}
        { mark-ref } {} { mini-env  } {} { mini-sep } {} { mini-right } {}
        { mini-right* } {}
      }
      { , { \exp_not:n {#1} } = { \exp_not:n {#2} } }
  }
```

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

11.26.4 Function for storing content in prop list

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

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

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

```
\cs_new_protected:Npn \__enumext_store_addto_prop:n #1
  {
    \prop_gput_if_not_in:cen { g__enumext_ \l__enumext_store_name_tl _prop }
      {
        \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } + 1 }
      }
      { #1 }
  }
\cs_generate_variant:Nn \__enumext_store_addto_prop:n { V, e }
```

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

11.26.5 Function for storing content in sequence

__enumext_store_addto_seq:n
__enumext_store_addto_seq:v
__enumext_store_addto_seq:V

The function __enumext_store_addto_seq:n stores the content in ⟨*sequence*⟩ defined by save-ans key. This function is used by \anskey in enumext, \item* in keyans and \anspic in keyanspic. The form in which the content is stored in ⟨*sequence*⟩ is in a internal enumext or enumext* environments with the *same structure* in which the command was executed.
The *"stored content"* is retrieved by means of the \printkeyans command.

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

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

11.26.6 Functions for storing the list structure in the sequence

__enumext_store_level_open:
__enumext_store_level_close:

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

```
\cs_new_protected:Nn \__enumext_store_level_open:
  {
    \bool_if:NT \l__enumext_check_answers_bool
      {
        \tl_if_empty:cTF { l__enumext_store_save_key_ \__enumext_level: _tl }
          {
            \__enumext_store_addto_seq:n
              {
                \item \begin{enumext}
              }
          }
          {
            \tl_put_left:cn { l__enumext_store_save_key_ \__enumext_level: _tl }
              {
                \item \begin{enumext} [
              }
            \tl_put_right:cn { l__enumext_store_save_key_ \__enumext_level: _tl }
              {
                ]
              }
            \__enumext_store_addto_seq:v { l__enumext_store_save_key_ \__enumext_level: _tl }
          }
      }
  }
\cs_new_protected:Nn \__enumext_store_level_close:
  {
    \bool_if:NT \l__enumext_check_answers_bool
      {
        \__enumext_store_addto_seq:n { \end{enumext} }
      }
  }
```

(*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.

```
\cs_new_protected:Nn \__enumext_store_level_open_vii:
  {
    \bool_if:NT \l__enumext_check_answers_bool
      {
        \tl_if_empty:NTF \l__enumext_store_save_key_vii_tl
          {
            \__enumext_store_addto_seq:n
              {
                \item \begin{enumext*}
              }
          }
          {
            \tl_put_left:Nn \l__enumext_store_save_key_vii_tl
              {
                \item \begin{enumext*}[
              }
            \tl_put_right:Nn \l__enumext_store_save_key_vii_tl
```

```
                {
                  ]
                }
              \__enumext_store_addto_seq:V \l__enumext_store_save_key_vii_tl
            }
        }
    }
  \cs_new_protected:Nn \__enumext_store_level_close_vii:
    {
      \bool_if:NT \l__enumext_check_answers_bool
        {
          \__enumext_store_addto_seq:n { \end{enumext*} }
        }
    }
```

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

11.26.7 Function for show marks and position

`\__enumext_print_keyans_box:NN`
`\__enumext_print_keyans_box:cc`

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

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

```
  \cs_new_protected:Nn \__enumext_print_keyans_box:NN
    {
      \mode_leave_vertical:
      \skip_horizontal:n { -\dim_use:N #2 }
      \makebox[0pt][ r ]
        {
          \makebox[ \dim_use:N #1 ][ \l__enumext_mark_position_str ]
            {
              \tl_use:N \l__enumext_mark_answer_sym_tl
            }
        }
      \skip_horizontal:n { \dim_use:N #2 }
    }
  \cs_generate_variant:Nn \__enumext_print_keyans_box:NN { cc }
```

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

11.27 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`.

```
  \cs_new_protected:Nn \__enumext_store_internal_ref:
    {
      \cs_set_protected:Npn \__enumext_tmp:n ##1
        {
          \tl_set_eq:cc { l__enumext_label_copy_##1_tl } { l__enumext_label_##1_tl }
          \tl_reverse:c { l__enumext_label_copy_##1_tl }
          \tl_remove_once:cn { l__enumext_label_copy_##1_tl } { . }
          \tl_reverse:c { l__enumext_label_copy_##1_tl }
        }
      \clist_map_inline:nn { i, ii, iii, iv, vii } { \__enumext_tmp:n {##1} }
      \cs_set:Npn \__enumext_tmp:n ##1
        { . \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.

```
      \bool_lazy_all:nT
        {
          { \bool_if_p:N \g__enumext_starred_bool }
          { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
        }
        {
          \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
          { \tl_use:N \l__enumext_label_copy_vii_tl }
      }
    \bool_lazy_all:nT
      {
        { \bool_if_p:N \l__enumext_standar_bool }
        { \bool_if_p:N \g__enumext_starred_bool }
        { \int_compare_p:nNn { \l__enumext_level_int } > { 0 } }
      }
      {
        \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
          {
            \tl_use:N \l__enumext_label_copy_vii_tl
            \int_step_function:nnN { 1 } { \l__enumext_level_int } \__enumext_tmp:n
          }
      }
```

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.

```
    \bool_lazy_all:nT
      {
        { \bool_if_p:N \l__enumext_standar_bool }
        { \int_compare_p:nNn { \l__enumext_level_int } > { 0 } }
        { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
        { \bool_not_p:n { \l__enumext_starred_bool } }
      }
      {
        \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
          {
            \tl_use:N \l__enumext_label_copy_i_tl
            \int_step_function:nnN { 2 } { \l__enumext_level_int } \__enumext_tmp:n
          }
      }
    \cs_set:Npn \__enumext_tmp:n ##1
      { \tl_use:c { l__enumext_label_copy_ \int_to_roman:n {##1} _tl } }
    \bool_lazy_all:nT
      {
        { \bool_if_p:N \l__enumext_standar_bool }
        { \int_compare_p:nNn { \l__enumext_level_int } > { 0 } }
        { \bool_not_p:n { \g__enumext_starred_bool } }
        { \int_compare_p:nNn { \l__enumext_level_h_int } > { 0 } }
      }
      {
        \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
          {
            \int_step_function:nnN { 1 } { \l__enumext_level_int } \__enumext_tmp:n
            . \tl_use:N \l__enumext_label_copy_vii_tl
          }
      }
```

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

```
    \tl_put_right:Ne \l__enumext_newlabel_arg_one_tl
      {
        \l__enumext_store_name_tl \c_colon_str
        \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } }
      }
```

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.

```
    \tl_put_right:Ne \l__enumext_write_aux_file_tl
      {
        \__enumext_newlabel:nn
          { \exp_not:V \l__enumext_newlabel_arg_one_tl }
          { \l__enumext_newlabel_arg_two_tl }
      }
    \l__enumext_write_aux_file_tl
  }
```

(*End of definition for* __enumext_store_internal_ref:.)

11.28 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*⟩.

```
\cs_new_protected:Npn \__enumext_store_anskey_code:n #1
  {
    \int_gincr:N \g__enumext_item_anskey_int
    \__enumext_store_addto_prop:n {#1}
    \bool_if:NT \l__enumext_store_ref_key_bool
      {
        \__enumext_store_internal_ref:
      }
    \__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.

```
    \tl_clear:N \l__enumext_store_anskey_arg_tl
    \bool_lazy_and:nnT
      { \bool_if_p:N \l__enumext_store_columns_break_bool }
      { \bool_not_p:n { \l__enumext_starred_bool } }
      {
        \tl_put_left:Nn \l__enumext_store_anskey_arg_tl { \columnbreak }
      }
    \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.

```
    \bool_lazy_and:nnT
      { \bool_not_p:n { \l__enumext_starred_bool } }
      { \int_compare_p:nNn { \l__enumext_store_item_join_int } > { 1 } }
      {
        \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
          {
            ( \exp_not:V \l__enumext_store_item_join_int )
          }
       }
```

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*.

```
     \bool_if:NTF \l__enumext_store_item_star_bool
      {
        \tl_put_right:Nn \l__enumext_store_anskey_arg_tl { * }
        \tl_if_empty:NF \l__enumext_store_item_symbol_tl
          {
            \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
              {
                [ \exp_not:V \l__enumext_store_item_symbol_tl ]
              }
          }
        \dim_compare:nT
          {
            \l__enumext_store_item_symbol_sep_dim != \c_zero_dim
          }
          {
            \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
              {
                [ \exp_not:V \l__enumext_store_item_symbol_sep_dim ]
              }
          }
        \tl_put_right:Nn \l__enumext_store_anskey_arg_tl {#1}
      }
      {
        \tl_put_right:Nn \l__enumext_store_anskey_arg_tl {#1}
      }
```

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*⟩.

```
    \bool_lazy_and:nnT
```

```
      { \bool_if_p:N \l__enumext_store_ref_key_bool }
      { \bool_if_p:N \l__enumext_hyperref_bool }
      {
        \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
          {
            \hfill \exp_not:N \hyperlink { \exp_not:V \l__enumext_newlabel_arg_one_tl }
                 { \exp_not:V \l__enumext_mark_ref_sym_tl }
          }
      }
    \__enumext_store_addto_seq:V \l__enumext_store_anskey_arg_tl
  }
```

(*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.

```
\cs_new_protected:Npn \__enumext_anskey_show_wrap_arg:n #1
  {
    \par
    \bool_if:NT \l__enumext_starred_bool
      {
       \cs_set:Nn \__enumext_level: { vii }
      }
    \__enumext_print_keyans_box:cc
      { l__enumext_labelwidth_ \__enumext_level: _dim }
      { l__enumext_labelsep_ \__enumext_level: _dim }
    \__enumext_anskey_wrapper:n { #1 }
  }
```

(*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.

```
\cs_new_protected:Npn \__enumext_anskey_show_wrap_left:n #1
  {
    \bool_if:NT \l__enumext_show_answer_bool
      {
        \__enumext_anskey_show_wrap_arg:n { #1 }
      }
    \bool_if:NT \l__enumext_show_position_bool
      {
        \tl_set:Ne \l__enumext_mark_answer_sym_tl
          {
            \group_begin:
            \exp_not:N \normalfont
            \exp_not:N \footnotesize [ \int_eval:n
              {
                \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
              }
              ]
            \group_end:
          }
        \__enumext_anskey_show_wrap_arg:n { #1 }
      }
  }
```

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

11.29 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*`.

```
\keys_define:nn { enumext / anskey }
  {
    break-col .bool_set:N = \l__enumext_store_columns_break_bool,
    break-col .default:n  = true,
    break-col .value_forbidden:n = true,
    item-join .int_set:N  = \l__enumext_store_item_join_int,
    item-join .value_required:n = true,
    item-star .bool_set:N = \l__enumext_store_item_star_bool,
    item-star .default:n  = true,
    item-star .value_forbidden:n = true,
    item-sym* .tl_set:N   = \l__enumext_store_item_symbol_tl,
    item-sym* .value_required:n = true,
    item-pos* .dim_set:N  = \l__enumext_store_item_symbol_sep_dim,
    item-pos* .value_required:n = true,
    unknown   .code:n     = { \__enumext_anskey_unknown:n {#1} },
  }
```

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`.

```
\cs_new_protected:Npn \__enumext_anskey_unknown:n #1
  {
    \exp_args:NV \__enumext_anskey_unknown:nn \l_keys_key_str {#1}
  }
\cs_new_protected:Npn \__enumext_anskey_unknown:nn #1 #2
  {
    \tl_if_blank:nTF {#2}
      {
        \msg_error:nnn { enumext } { anskey-cmd-key-unknown } {#1}
      }
      {
        \msg_error:nnnn { enumext } { anskey-cmd-key-value-unknown } {#1} {#2}
      }
  }
```

(*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`.

```
\NewDocumentCommand \anskey { o +m }
  {
    \__enumext_anskey_safe_outer:
    \group_begin:
      \bool_if:NT \l__enumext_check_answers_bool
        {
          \tl_if_novalue:nF {#1}
            {
              \keys_set:nn { enumext / anskey } {#1}
            }
          \tl_if_blank:nTF {#2}
            {
              \msg_error:nn { enumext } { anskey-empty-arg }
            }
            {
              \__enumext_anskey_safe_inner:
              \__enumext_store_anskey_code:n {#2}
            }
        }
    \group_end:
  }
```

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

11.29.1 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.

```
\cs_new_protected:Nn \__enumext_anskey_safe_outer:
  {
    \bool_if:NF \l__enumext_store_active_bool
```

```
      {
        \msg_error:nnnn { enumext } { anskey-wrong-place }{ anskey }{ enumext }
      }
    \int_compare:nNnT { \l__enumext_keyans_level_int } = { 1 }
      {
        \msg_error:nnnn { enumext } { command-wrong-place }{ anskey }{ keyans }
      }
    \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
      {
        \msg_error:nnnn { enumext } { command-wrong-place }{ anskey }{ keyans* }
      }
    \int_compare:nNnT { \l__enumext_keyans_pic_level_int } = { 1 }
      {
        \msg_error:nnnn { enumext } { command-wrong-place }{ anskey }{ keyanspic }
      }
  }
```

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.

```
\cs_new_protected:Nn \__enumext_anskey_safe_inner:
  {
    \int_incr:N \l__enumext_anskey_level_int
    \int_compare:nNnT { \l__enumext_anskey_level_int } > { 1 }
      {
        \msg_error:nn { enumext } { anskey-nested }
      }
    \bool_if:NF \l__enumext_item_number_bool
      {
        \msg_error:nn { enumext } { anskey-unnumber-item }
      }
    \mode_if_math:T
      {
        \msg_error:nne { enumext } { anskey-math-mode } { \c_backslash_str anskey }
      }
  }
```

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

11.30 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:` (§11.31) which is executed after the environment in which the key `save-ans` is active.

```
\cs_new_protected:Nn \__enumext_undefine_anskey_env:
  {
    \cs_undefine:c { anskey* }
    \cs_undefine:c { endanskey* }
    \cs_undefine:c { __scontents_anskey*_env_begin: }
    \cs_undefine:c { __scontents_anskey*_env_end: }
  }
```

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

```
\__enumext_before_env:nn { enumext }
  {
    \bool_lazy_and:nnT
      { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
      { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
      {
        \cs_if_free:cF { __scontents_anskey*_env_begin: }
          {
            \msg_error:nnn { enumext } { anskey-env-error } { anskey* }
          }
      }
  }
\__enumext_before_env:nn { enumext* }
  {
    \bool_lazy_and:nnT
      { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
      { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
      {
        \cs_if_free:cF { __scontents_anskey*_env_begin: }
          {
            \msg_error:nnn { enumext } { anskey-env-error } { anskey* }
          }
      }
  }
```

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.

```
\__enumext_before_env:nn { anskey* }
  {
    \int_compare:nNnT { \l__enumext_keyans_level_int } = { 1 }
      {
        \msg_error:nnn { enumext } { anskey-env-wrong }{ keyans }
      }
    \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
      {
        \msg_error:nnn { enumext } { anskey-env-wrong } { keyans* }
      }
    \int_compare:nNnT { \l__enumext_keyans_pic_level_int } = { 1 }
      {
        \msg_error:nnn { enumext } { anskey-env-wrong } { keyanspic }
      }
    \bool_if:NF \l__enumext_item_number_bool
      {
        \msg_error:nn { enumext } { anskey-unnumber-item }
      }
    \mode_if_math:T
      {
        \msg_error:nnn { enumext } { anskey-math-mode } { anskey* }
      }
  }
```

(*End of definition for* __enumext_undefine_anskey_env:.)

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

The function __enumext_anskey_env_make:n creates the environment anskey* (*custom version* of scontents environment) by setting the initial keys store-env={⟨*store name*⟩} and print-env=false. To maintain the *scope* of the environment and that it is only active when the key save-ans is active we will pass this function to the function __enumext_storing_exec: (§11.25.1) 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.

```
\cs_new_protected:Npn \__enumext_anskey_env_make:n #1
  {
    \bool_if:NT \l__enumext_anskey_env_bool
      {
        \newenvsc{anskey*}[store-env=#1,print-env=false]
        \__enumext_anskey_env_exec:
      }
  }
\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.

```
\cs_new_protected:Nn \__enumext_anskey_env_define_keys:
  {
    \keys_define:nn { scontents / scontents }
      {
        break-col .bool_gset:N = \g__enumext_store_columns_break_bool,
        break-col .default:n   = true,
        break-col .value_forbidden:n = true,
        item-join .int_gset:N  = \g__enumext_store_item_join_int,
        item-join .value_required:n = true,
        item-star .bool_gset:N = \g__enumext_store_item_star_bool,
        item-star .default:n   = true,
        item-star .value_forbidden:n = true,
        item-sym* .tl_gset:N   = \g__enumext_store_item_symbol_tl,
        item-sym* .value_required:n = true,
        item-pos* .dim_gset:N  = \g__enumext_store_item_symbol_sep_dim,
        item-pos* .value_required:n = true,
        print-env .undefine:,
        store-env .undefine:,
        write-out .undefine:,
        unknown   .code:n      = { \__enumext_anskey_env_unknown:n {##1} },
      }
  }
```

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`.

```
\cs_new_protected:Npn \__enumext_anskey_env_unknown:n #1
  {
    \exp_args:NV \__enumext_anskey_env_unknown:nn \l_keys_key_str {#1}
  }
\cs_new_protected:Npn \__enumext_anskey_env_unknown:nn #1#2
  {
    \tl_if_blank:nTF {#2}
      {
        \msg_error:nnn { enumext } { anskey-env-key-unknown } {#1}
      }
      {
        \msg_error:nnnn { enumext } { anskey-env-key-value-unknown } {#1} {#2}
      }
  }
```

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`.

```
\cs_new_protected:Nn \__enumext_anskey_env_reset_keys:
  {
    \keys_define:nn { scontents / scontents }
      {
        break-col .undefine:,
        item-join .undefine:,
        item-star .undefine:,
        item-sym* .undefine:,
        item-pos* .undefine:,
        write-out .code:n    = {
                                 \bool_set_false:N \l__scontents_storing_bool
                                 \bool_set_true:N  \l__scontents_writing_bool
                                 \tl_set:Nn \l__scontents_fname_out_tl {##1}
                               },
        write-out .value_required:n = true,
        print-env .meta:nn   = { scontents } { print-env = ##1 },
        print-env .default:n = true,
        store-env .meta:nn   = { scontents } { store-env = ##1 },
        unknown   .code:n    = { \__scontents_parse_environment_keys:n {##1} },
      }
  }
```

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*.

```
\cs_new_protected:Npn \__enumext_rescan_anskey_env:n #1
  {
    \group_begin:
      \int_set:Nn \tex_newlinechar:D { `\^^J }
      \__scontents_rescan_tokens:x
        {
          \endgroup % This assumes \catcode`\\=0... Things might go off otherwise.
         #1
        }
  }
```

(*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*⟩.

```
\cs_new_protected:Nn \__enumext_anskey_env_exec:
  {
    \__enumext_before_env:nn { anskey* }
      {
        \__enumext_anskey_env_define_keys:
      }
```

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.

```
    \hook_if_empty:nF {env/anskey*/after}
      {
        \hook_gremove_code:nn {env/anskey*/after} { * }
      }
    \__enumext_after_env:nn { anskey* }
      {
        \__enumext_anskey_env_save_keys:
        \tl_clear:N \l__enumext_store_anskey_env_tl
        \tl_clear:N \l__enumext_store_anskey_opt_tl
        \bool_if:NT \l__enumext_check_answers_bool
          {
            \tl_gset:Ne \l__enumext_store_anskey_env_tl
              {
                \seq_item:ce { g__scontents_name_ \l__enumext_store_name_tl _seq } { -1 }
              }
            \regex_match:nVTF
              { ^\s* \z | ^\s* \u{c__scontents_hidden_space_str} \z }
              \l__enumext_store_anskey_env_tl
              {
                \msg_error:nn { enumext } { anskey-empty-arg }
              }
              {
                \__enumext_anskey_env_store:
              }
          }
        \__enumext_anskey_env_clean_vars:
        \__enumext_anskey_env_reset_keys:
      }
  }
```

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`.

```
\cs_new_protected:Nn \__enumext_anskey_env_save_keys:
  {
    \bool_lazy_and:nnT
      { \bool_if_p:N \g__enumext_store_columns_break_bool }
      { \bool_not_p:n { \l__enumext_starred_bool } }
      {
        \tl_put_left:Ne \l__enumext_store_anskey_opt_tl { ,break-col, }
      }
```

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

```
    \bool_lazy_and:nnT
      { \bool_not_p:n { \l__enumext_starred_bool } }
      { \int_compare_p:nNn { \g__enumext_store_item_join_int } > { 1 } }
      {
        \tl_put_left::Ne \l__enumext_store_anskey_opt_tl
          {
            ,item-join = \exp_not:V \g__enumext_store_item_join_int,
          }
       }
```

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

```
    \bool_if:NT \g__enumext_store_item_star_bool
      {
        \tl_put_left:Ne \l__enumext_store_anskey_opt_tl
          {
            ,item-star,
          }
        \tl_if_empty:NF \g__enumext_store_item_symbol_tl
          {
            \tl_put_left:Ne \l__enumext_store_anskey_opt_tl
              {
                ,item-sym* = \exp_not:V \g__enumext_store_item_symbol_tl,
              }
          }
        \dim_compare:nT
          {
            \g__enumext_store_item_symbol_sep_dim != \c_zero_dim
          }
          {
            \tl_put_left:Ne \l__enumext_store_anskey_opt_tl
              {
                ,item-pos* = \exp_not:V \g__enumext_store_item_symbol_sep_dim,
              }
           }
       }
  }
```

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`.

```
\cs_new_protected:Nn \__enumext_anskey_env_store:
  {
    \group_begin:
      \tl_if_empty:NTF \l__enumext_store_anskey_opt_tl
        {
          \exp_args:Ne
            \__enumext_store_anskey_code:n
              {
                \__enumext_rescan_anskey_env:n { \l__enumext_store_anskey_env_tl }
              }
        }
        {
          \keys_set_known:nV { enumext / anskey } \l__enumext_store_anskey_opt_tl
          \exp_args:Ne
            \__enumext_store_anskey_code:n
              {
                \__enumext_rescan_anskey_env:n { \l__enumext_store_anskey_env_tl }
              }
        }
    \group_end:
  }
```

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

```
\cs_new_protected:Nn \__enumext_anskey_env_clean_vars:
  {
    \bool_gset_false:N \g__enumext_store_columns_break_bool
    \int_gzero:N       \g__enumext_store_item_join_int
    \bool_gset_false:N \g__enumext_store_item_star_bool
    \tl_gclear:N       \g__enumext_store_item_symbol_tl
    \dim_gzero:N       \g__enumext_store_item_symbol_sep_dim
  }
```

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

11.31 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* (§11.30) 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.

```
\cs_new_protected:Nn \__enumext_execute_after_env:
  {
    \int_compare:nNnT { \l__enumext_level_int } = { 0 }
      {
        \tl_if_empty:NF \g__enumext_store_name_tl
          {
            \__enumext_stop_save_ans_msg:
            \__enumext_item_answer_diff:
            \__enumext_log_global_vars:
            \__enumext_log_answer_vars:
            \bool_if:NTF \g__enumext_check_ans_key_bool
              {
                \__enumext_check_ans_show:
              }
              { \__enumext_check_ans_log: }
            \__enumext_undefine_anskey_env:
          }
        \__enumext_reset_global_vars:
      }
  }
```

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

This function is passed to the function __enumext_after_env:nn for the environments enumext (§11.38) and enumext* (§11.42) and it is executed only when the environments are not nested or at some level of these..

11.32 Common functions for keyans, keyans* and keyanspic

11.32.1 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.

```
\cs_new_protected:Npn \__enumext_keyans_addto_prop:n #1
  {
    \tl_clear:N \l__enumext_store_current_label_tl
    \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
      {
        \tl_put_right:Ne \l__enumext_store_current_label_tl { \l__enumext_label_vi_tl }
      }
      {
        \tl_put_right:Ne \l__enumext_store_current_label_tl { \l__enumext_label_v_tl }
      }
    \tl_if_novalue:nF { #1 }
      {
        % Set save-sep
        \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_tl
          {
            \tl_put_right:Ne \l__enumext_store_current_label_tl { \l__enumext_store_keyans_item_op
          }
        \tl_put_right:Ne \l__enumext_store_current_label_tl { #1 }
      }
    \__enumext_store_addto_prop:V \l__enumext_store_current_label_tl
  }
```

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

11.32.2 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.

```
\cs_new_protected:Nn \__enumext_keyans_store_ref:
  {
    \bool_if:NT \l__enumext_store_ref_key_bool
      {
        \cs_set_protected:Npn \__enumext_tmp:n ##1
          {
            \tl_set_eq:cc { l__enumext_label_copy_##1_tl } { l__enumext_label_##1_tl }
            \tl_reverse:c { l__enumext_label_copy_##1_tl }
            \tl_remove_once:cn { l__enumext_label_copy_##1_tl } { . }
            \tl_reverse:c { l__enumext_label_copy_##1_tl }
          }
        \clist_map_inline:nn { i, v, vi, vii, viii } { \__enumext_tmp:n {##1} }
        \__enumext_keyans_store_ref_aux_i:
      }
   }
```

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.

```
\cs_new_protected:Nn \__enumext_keyans_store_ref_aux_i:
  {
    \bool_if:NT \g__enumext_starred_bool
      {
        \tl_set_eq:NN \l__enumext_label_copy_i_tl \l__enumext_label_copy_vii_tl
      }
    \int_compare:nNnT { \l__enumext_keyans_pic_level_int } = { 1 }
      {
        \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
          { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_vi_tl }
      }
    \int_compare:nNnT { \l__enumext_keyans_level_int } = { 1 }
      {
        \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
          { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_v_tl }
      }
    \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
      {
        \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
          { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_viii_tl }
      }
    \tl_put_right:Ne \l__enumext_newlabel_arg_one_tl
      {
        \l__enumext_store_name_tl \c_colon_str
        \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } }
      }
     \__enumext_keyans_store_ref_aux_ii:
  }
```

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.

```
\cs_new_protected:Nn \__enumext_keyans_store_ref_aux_ii:
  {
    \tl_put_right:Ne \l__enumext_write_aux_file_tl
      {
        \__enumext_newlabel:nn
          { \exp_not:V \l__enumext_newlabel_arg_one_tl }
          { \l__enumext_newlabel_arg_two_tl }
      }
    \l__enumext_write_aux_file_tl
  }
```

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

11.32.3 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.

```
\cs_new_protected:Npn \__enumext_keyans_addto_seq:n #1
  {
    \tl_clear:N \l__enumext_store_current_label_tl
    \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
      {
        \tl_put_right:Ne \l__enumext_store_current_label_tl { \item \l__enumext_label_vi_tl }
      }
      {
        \tl_put_right:Ne \l__enumext_store_current_label_tl { \item \l__enumext_label_v_tl }
      }
    \tl_if_novalue:nF { #1 }
      {
        \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_tl
          {
            \tl_put_right:Ne \l__enumext_store_current_label_tl
              {
                \l__enumext_store_keyans_item_opt_sep_tl
              }
          }
        \tl_put_right:Ne \l__enumext_store_current_label_tl { #1 }
      }
    \__enumext_keyans_addto_seq_link:
  }
```

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.

```
\cs_new_protected:Nn \__enumext_keyans_addto_seq_link:
  {
    \bool_lazy_and:nnT
      { \bool_if_p:N \l__enumext_store_ref_key_bool }
      { \bool_if_p:N \l__enumext_hyperref_bool }
      {
        \tl_put_right:Ne \l__enumext_store_current_label_tl
          {
            \hfill \exp_not:N \hyperlink
              {
                \exp_not:V \l__enumext_newlabel_arg_one_tl
              }
              { \exp_not:V \l__enumext_mark_ref_sym_tl }
          }
      }
    \__enumext_store_addto_seq:V \l__enumext_store_current_label_tl
    \bool_if:NT \l__enumext_check_answers_bool
      {
        \int_gincr:N \g__enumext_item_anskey_int
      }
  }
```

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

11.32.4 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.

```
\cs_new_protected:Npn \__enumext_keyans_show_left:n #1
  {
    \tl_if_novalue:nF { #1 }
      {
        \tl_set:Ne \l__enumext_store_current_opt_arg_tl { #1 }
      }
    \bool_if:NT \l__enumext_show_answer_bool
      {
        \__enumext_keyans_show_ans:
      }
    \bool_if:NT \l__enumext_show_position_bool
      {
        \__enumext_keyans_show_pos:
      }
  }
\cs_new_protected:Nn \__enumext_keyans_show_item_opt:
  {
    \tl_if_empty:NF \l__enumext_store_current_opt_arg_tl
      {
        \bool_lazy_or:nnT
          { \bool_if_p:N \l__enumext_show_answer_bool }
          { \bool_if_p:N \l__enumext_show_position_bool }
          {
            \__enumext_keyans_wrapper_opt:n { \l__enumext_store_current_opt_arg_tl } \c_space_tl
          }
      }
  }
\cs_new_protected:Nn \__enumext_keyans_show_ans:
  {
    \tl_put_left:Nn \l__enumext_label_v_tl
      {
        \__enumext_print_keyans_box:NN
          \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
      }
  }
\cs_new_protected:Nn \__enumext_keyans_show_pos:
  {
    \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
      {
        \tl_set:Ne \l__enumext_mark_answer_sym_tl
          {
            \group_begin:
            \exp_not:N \normalfont
            \exp_not:N \footnotesize [ \int_eval:n
              {
                \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
              }
              ]
            \group_end:
          }
      }
      {
        \tl_set:Ne \l__enumext_mark_answer_sym_tl
          {
            \group_begin:
            \exp_not:N \normalfont
            \exp_not:N \footnotesize [ \int_eval:n
              {
                \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } + 1
              }
              ]
            \group_end:
          }
      }
    \tl_put_left:Nn \l__enumext_label_v_tl
      {
        \__enumext_print_keyans_box:NN
          \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
      }
  }
```

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

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

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

item-sym*
item-pos*

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

```
\cs_set_protected:Npn \__enumext_tmp:nn #1 #2
  {
    \keys_define:nn { enumext / #1 }
      {
        item-sym* .tl_set:c  = { l__enumext_item_symbol_#2_tl },
        item-sym* .value_required:n = true,
        item-sym* .initial:n  = {$\star$},
        item-pos* .dim_set:c  = { l__enumext_item_symbol_sep_#2_dim },
        item-pos* .value_required:n = true,
      }
  }
\clist_map_inline:nn
  {
    {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {enumext*}{vii}
  }
  { \__enumext_tmp:nn #1 }
```

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

11.34 Redefining \footnote command

__enumext_footnotetext:nn
__enumext_renew_footnote:
__enumext_print_footnote:

To keep the correct numbering of \footnote and to make it work correctly with the mini-env key and in the enumext* and keyans* environments, it is necessary to redefine the command. This implementation is adapted from the answer given by Clea F. Rees (@cfr) in footnotes in boxes compatible with hyperref.

```
\cs_new_protected:Nn \__enumext_footnotetext:nn
  {
    \footnotetext[#1]{#2}
  }
\cs_new_protected:Nn \__enumext_renew_footnote:
  {
    \seq_gclear:N \g__enumext_footnote_arg_seq
    \seq_gclear:N \g__enumext_footnote_int_seq
    \RenewDocumentCommand \footnote { o +m }
      {
        \tl_if_novalue:nTF {##1}
          {
            \stepcounter{footnote}
            \int_gset_eq:Nc \g__enumext_footnote_int { c@footnote }
          }
          {
            \int_gset:Nn \g__enumext_footnote_int { ##1 }
          }
        \footnotemark [ \g__enumext_footnote_int ]
        \seq_gput_right:Nn \g__enumext_footnote_arg_seq { ##2 }
        \seq_gput_right:NV \g__enumext_footnote_int_seq \g__enumext_footnote_int
     }
  }
\cs_new_protected:Nn \__enumext_print_footnote:
  {
    \seq_if_empty:NF \g__enumext_footnote_int_seq
      {
        \seq_map_pairwise_function:NNN
          \g__enumext_footnote_int_seq
          \g__enumext_footnote_arg_seq
          \__enumext_footnotetext:nn
      }
  }
```

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

11.35 Redefining \item command

Redefining the \item command is not as simple as I thought. This command works in conjunction with the \makelabel command so I have to redefine both of them, in addition to this, we will have to use a couple of *global* variables to pass the values from one command to the other.

11.35.1 The \item command in enumext

__enumext_default_item:n

The \item and \item[⟨custom⟩] commands work in the usual way on enumext.

First we will see if the optional argument is present, if it is NOT present we will check the state of the variable \l__enumext_check_ans_key_bool set by the key check-ans, set the boolean variable \l__enumext_wrap_label_X_bool to "true" and execute __enumext_item_std:w.

Otherwise we will check the state of the boolean variable \l__enumext_wrap_label_opt_X_bool set by the key wrap-label* and execute __enumext_item_std:w with the optional argument.

The boolean variable \l__enumext_wrap_label_X_bool is used by the function __enumext_make_-label: (§11.36).

```
\cs_new_protected:Npn \__enumext_default_item:n #1
  {
    \tl_if_novalue:nTF {#1}
      {
        \bool_if:NT \l__enumext_check_answers_bool
          {
            \int_gincr:N \g__enumext_item_number_int
            \bool_set_true:N \l__enumext_item_number_bool
          }
        \bool_set_true:c { l__enumext_wrap_label_ \__enumext_level: _bool }
        \__enumext_item_std:w \tl_use:c { l__enumext_fake_item_indent_ \__enumext_level: _tl }
      }
      {
        \bool_set_eq:cc
          { l__enumext_wrap_label_ \__enumext_level: _bool }
          { l__enumext_wrap_label_opt_ \__enumext_level: _bool }
        \__enumext_item_std:w [#1] \tl_use:c { l__enumext_fake_item_indent_ \__enumext_level: _tl
      }
  }
```

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

__enumext_starred_item:nn

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

Then we will see the state of the variable \l__enumext_check_ans_key_bool set by the key check-ans, set the boolean variable \l__enumext_wrap_label_X_bool to "true" and execute __enumext_item_-std:w.

In this function the optional argument of __enumext_item_std:w is omitted, we only want it to be numbered.

The boolean variable \l__enumext_wrap_label_X_bool and the vars \l__enumext_item_symbol_-sep_X_dim, \g__enumext_item_symbol_tl are used by the function __enumext_make_label: (§11.36).

```
\cs_new_protected:Npn \__enumext_starred_item:nn #1 #2
  {
    \tl_if_novalue:nF {#1}
      {
        \tl_set:cn { l__enumext_item_symbol_ \__enumext_level: _tl } {#1}
      }
    \tl_gset_eq:Nc \g__enumext_item_symbol_tl { l__enumext_item_symbol_ \__enumext_level: _tl }
    \tl_if_novalue:nTF {#2}
      {
        \dim_set_eq:cc
          { l__enumext_item_symbol_sep_ \__enumext_level: _dim }
          { l__enumext_labelsep_ \__enumext_level: _dim }
      }
      {
        \dim_set:cn { l__enumext_item_symbol_sep_ \__enumext_level: _dim } {#2}
      }
    \bool_if:NT \l__enumext_check_answers_bool
      {
        \int_gincr:N \g__enumext_item_number_int
        \bool_set_true:N \l__enumext_item_number_bool
      }
    \bool_set_true:c { l__enumext_wrap_label_ \__enumext_level: _bool }
    \__enumext_item_std:w \tl_use:c { l__enumext_fake_item_indent_ \__enumext_level: _tl }
  }
```

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

`\__enumext_redefine_item:`

The function `\__enumext_redefine_item:` will redefine the `\item` command in the `enumext` environment for the internal mechanism of check-answers for `check-ans` key and adding the starred `\item*` version.

This function is passed to `\__enumext_list_arg_two_X:` which is used in the definition of the `enumext` environment (§11.37.2).

```
\cs_new_protected:Nn \__enumext_redefine_item:
  {
    \RenewDocumentCommand \item { s o o }
      {
       \bool_if:nTF {##1}
         {
           \__enumext_starred_item:nn {##2} {##3}
         }
         { \__enumext_default_item:n {##2} }
      }
  }
```

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

11.35.2 The `\item` command in keyans

The `\item*` and `\item*[⟨content⟩]` commands *store* the current ⟨*label*⟩ next to the [⟨*content*⟩] if it is present in the ⟨*sequence*⟩ and ⟨*prop list*⟩ defined by `save-ans` key.

`\__enumext_keyans_default_item:n`

The function `\__enumext_keyans_default_item:n` executes the original behavior of the `\item`.

```
\cs_new_protected:Npn \__enumext_keyans_default_item:n #1
  {
    \tl_if_novalue:nTF { #1 }
      {
        \bool_set_true:N \l__enumext_wrap_label_v_bool
        \__enumext_item_std:w \tl_use:N \l__enumext_fake_item_indent_v_tl
      }
      {
        \bool_set_eq:NN \l__enumext_wrap_label_v_bool \l__enumext_wrap_label_opt_v_bool
        \__enumext_item_std:w [#1] \tl_use:N \l__enumext_fake_item_indent_v_tl
      }
  }
```

(*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*⟩.

```
\cs_new_protected:Npn \__enumext_keyans_starred_item:n #1
  {
    \tl_set_eq:NN \l__enumext_store_current_label_tmp_tl \l__enumext_label_v_tl
    \__enumext_keyans_show_left:n { #1 }
    \bool_set_true:N \l__enumext_wrap_label_v_bool
    \__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*⟩.

```
    \tl_set_eq:NN \l__enumext_label_v_tl \l__enumext_store_current_label_tmp_tl
    \__enumext_keyans_addto_prop:n { #1 }
    \__enumext_keyans_store_ref:
    \__enumext_keyans_addto_seq:n { #1 }
    \int_gincr:N \g__enumext_check_starred_cmd_int
  }
```

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

\item*
__enumext_keyans_redefine_item:

The function __enumext_keyans_redefine_item: is responsible for adding the *starred* and *optional* argument by the __enumext_list_arg_two_v: function in the definition of the keyans environment. Here we need to use \peek_remove_spaces:n to prevent an unwanted space when using \item* in conjunction with the itemindent key.

This function is passed to __enumext_list_arg_two_v: which is used in the definition of the keyans environment (§11.37.2).

```
\cs_new_protected:Nn \__enumext_keyans_redefine_item:
  {
    \RenewDocumentCommand \item { s o }
      {
        \bool_if:nTF {##1}
          {
            \peek_remove_spaces:n
              {
                \__enumext_keyans_starred_item:n {##2}
              }
          }
          {
            \__enumext_keyans_default_item:n {##2}
          }
      }
  }
```

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

11.36 Redefining \makelabel command

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

11.36.1 Redefining \makelabel for enumext

__enumext_item_starred:

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

```
\cs_new_protected:Nn \__enumext_item_starred:
  {
    \tl_if_empty:cF { l__enumext_item_symbol_ \__enumext_level: _tl }
      {
        \mode_leave_vertical:
        \skip_horizontal:n { -\dim_use:c { l__enumext_item_symbol_sep_ \__enumext_level: _dim } }
        \makebox[ 0pt ][ r ]{ \g__enumext_item_symbol_tl }
        \skip_horizontal:n { \dim_use:c { l__enumext_item_symbol_sep_ \__enumext_level: _dim } }
      }
  }
```

(*End of definition for* __enumext_item_starred:.)

__enumext_make_label:

The function __enumext_make_label: redefine \makelabel for the enumext environment.

This function is passed to __enumext_list_arg_two_X: which is used in the definition of the enumext environment (§11.37.2).

```
\cs_new_protected:Nn \__enumext_make_label:
  {
    \RenewDocumentCommand \makelabel { m }
      {
        \tl_use:c { l__enumext_label_fill_left_ \__enumext_level: _tl }
        \tl_use:c { l__enumext_label_font_style_ \__enumext_level: _tl }
        \bool_if:cTF { l__enumext_wrap_label_ \__enumext_level: _bool }
          {
            \__enumext_item_starred:
            \use:c { __enumext_wrapper_label_ \__enumext_level: :n } { ##1 }
          }
          { ##1 }
        \tl_use:c { l__enumext_label_fill_right_ \__enumext_level: _tl }
        \tl_gclear:N \g__enumext_item_symbol_tl
      }
  }
```

(*End of definition for* __enumext_make_label:.)

11.36.2 Redefining \makelabel for keyans

__enumext_keyans_make_label: The function __enumext_keyans_make_label: redefine \makelabel for keyans environment.

This function is passed to __enumext_list_arg_two_v: which is used in the definition of the keyans environment (§11.37.2).

```
\cs_new_protected:Nn \__enumext_keyans_make_label:
  {
    \RenewDocumentCommand \makelabel { m }
      {
        \tl_use:N \l__enumext_label_fill_left_v_tl
        \tl_use:N \l__enumext_label_font_style_v_tl
        \bool_if:NTF \l__enumext_wrap_label_v_bool
          {
            \__enumext_wrapper_label_v:n { ##1 }
          }
          { ##1 }
        \tl_use:N \l__enumext_label_fill_right_v_tl
      }
  }
```

(*End of definition for* __enumext_keyans_make_label:.)

11.37 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.

11.37.1 Calculation of \leftmargin and \itemindent

Consider the figure 9 where the default margins (on the left) of a list are represented.

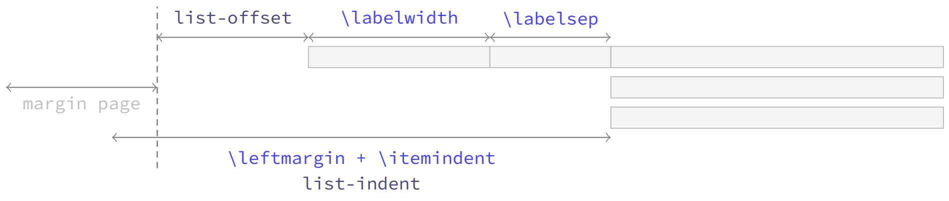


Figure 9: Representation of standard horizontal lengths in list environment.

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.

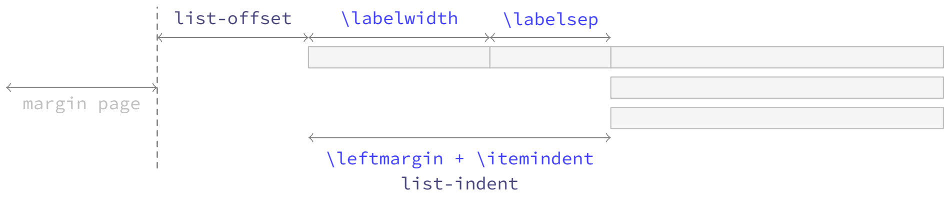


Figure 10: Representation of horizontal lengths concept in list in enumext.

Where the default values will look like in the figure 11.

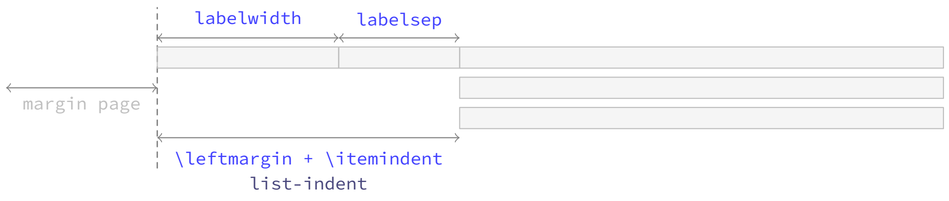


Figure 11: Default horizontal lengths in enumext.

__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 (§11.37.2).

```
\cs_new_protected:Npn \__enumext_calc_hspace:NNNNNNN #1 #2 #3 #4 #5 #6 #7
  {
    \dim_compare:nNnT { #1 } < { \c_zero_dim }
      {
        \msg_warning:nnnV { enumext } { width-non-positive }{ labelwidth }{ #1 }
        \dim_set:Nn #1 { \dim_abs:n { #1 } }
      }
    \dim_compare:nNnT { #2 } < { \c_zero_dim }
      {
        \msg_warning:nnnV { enumext } { width-negative }{ labelsep }{ #2 }
        \dim_set:Nn #2 { \dim_abs:n { #2 } }
      }
```

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

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

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

```
    \dim_compare:nNnTF { #4 } < { \c_zero_dim }
      {
        \dim_set:Nn #6 { #1 + #2 - #4}
        \dim_set:Nn #5 { #1 + #2 + #3 - #6 }
      }
      {
        \dim_compare:nNnT { #4 } = { #1 + #2 }
          { \dim_set:Nn #6 { \c_zero_dim } }
        \dim_compare:nNnT { #4 } < { #1 + #2 }
          { \dim_set:Nn #6 { #1 + #2 - #4} }
        \dim_compare:nNnT { #4 } > { #1 + #2 }
          {
            \dim_set:Nn #6 { -#1 - #2 + #4}
            \dim_set:Nn #6 { #6*-1}
          }
        \dim_set:Nn #5 { #1 + #2 + #3 - #6 }
      }
  }
\cs_generate_variant:Nn \__enumext_calc_hspace:NNNNNNN { ccccccc }
```

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

11.37.2 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.

```
\cs_set_protected:Npn \__enumext_tmp:n #1
  {
    \cs_new_protected:cpn { __enumext_list_arg_two_#1: }
      {
        \__enumext_calc_hspace:ccccccc
          { l__enumext_labelwidth_#1_dim } { l__enumext_labelsep_#1_dim }
          { l__enumext_listoffset_#1_dim } { l__enumext_leftmargin_tmp_#1_dim }
          { l__enumext_leftmargin_#1_dim } { l__enumext_itemindent_#1_dim }
          { l__enumext_leftmargin_tmp_#1_bool }
        \clist_map_inline:nn
          { labelsep, labelwidth, itemindent, leftmargin, rightmargin, listparindent }
          { \dim_set_eq:cc {####1} { l__enumext_####1_#1_dim } }
        \clist_map_inline:nn { topsep, parsep, partopsep, itemsep }
          { \skip_set_eq:cc {####1} { l__enumext_####1_#1_skip } }
        \usecounter { enumX#1 }
        \setcounter { enumX#1 } { \int_eval:n { \int_use:c { l__enumext_start_#1_int } - 1 } }
        \str_if_eq:nnTF {#1} { v }
          {
            \__enumext_keyans_redefine_item:
            \__enumext_keyans_make_label:
            \__enumext_keyans_ref:
            \__enumext_keyans_fake_item:
            \bool_if:cT { l__enumext_show_length_#1_bool }
              {
                \msg_term:nnnn { enumext } { list-lengths-not-nested } { v } { keyans }
              }
          }
          {
            \__enumext_redefine_item:
            \__enumext_make_label:
            \__enumext_standar_ref:
            \__enumext_fake_item:
            \bool_if:cT { l__enumext_show_length_#1_bool }
              {
                \msg_term:nnne { enumext } { list-lengths } {#1} { \int_use:N \l__enumext_level_i
              }
          }
      }
  }
\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.

```
\cs_set_protected:Npn \__enumext_tmp:n #1
  {
    \cs_new_protected:cpn { __enumext_list_arg_two_#1: }
      {
        \bool_set_true:c { l__enumext_leftmargin_tmp_#1_bool }
        \dim_zero:c { l__enumext_leftmargin_tmp_#1_dim }
        \__enumext_calc_hspace:ccccccc
          { l__enumext_labelwidth_#1_dim } { l__enumext_labelsep_#1_dim }
          { l__enumext_listoffset_#1_dim } { l__enumext_leftmargin_tmp_#1_dim }
          { l__enumext_leftmargin_#1_dim } { l__enumext_itemindent_#1_dim }
          { l__enumext_leftmargin_tmp_#1_bool }
        \clist_map_inline:nn
          { labelsep, labelwidth, itemindent, leftmargin, rightmargin, listparindent }
          { \dim_set_eq:cc {####1} { l__enumext_####1_#1_dim } }
        \clist_map_inline:nn { topsep, parsep, partopsep, itemsep }
          { \skip_set_eq:cc {####1} { l__enumext_####1_#1_skip } }
        \skip_set_eq:Nc \parsep  { l__enumext_itemsep_#1_skip }
        \skip_zero:N \partopsep
        \usecounter { enumX#1 }
        \setcounter { enumX#1 } { \int_eval:n { \int_use:c { l__enumext_start_#1_int } - 1 } }
        \__enumext_starred_ref:
        \str_if_eq:nnTF {#1} { vii }
          {
            \__enumext_fake_item_vii:
            \bool_if:cT { l__enumext_show_length_vii_bool }
              { \msg_term:nnnn { enumext } { list-lengths-not-nested } { vii } { enumext* } }
          }
          {
            \__enumext_fake_item_viii:
            \bool_if:cT { l__enumext_show_length_#1_bool }
              { \msg_term:nnnn { enumext } { list-lengths-not-nested } { #1 } { keyans* } }
          }
      }
  }
\clist_map_inline:nn { vii, viii } { \__enumext_tmp:n {#1} }
```

(*End of definition for* `\__enumext_list_arg_two_vii:` *and* `\__enumext_list_arg_two_viii:`*.*)

11.38 The environment enumext

`enumext`

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

```
\NewDocumentEnvironment{enumext}{ O{} }
  {
    \__enumext_safe_exec:
    \__enumext_parse_keys:n {#1}
    \__enumext_before_list:
    \__enumext_start_store_level:
    \__enumext_start_list:nn
      { \tl_use:c { l__enumext_label_ \__enumext_level: _tl } }
      {
        \use:c { __enumext_list_arg_two_ \__enumext_level: : }
        \__enumext_before_keys_exec:
      }
    \__enumext_after_args_exec:
  }
  {
    \__enumext_stop_list:
    \__enumext_stop_store_level:
    \__enumext_after_list:
  }
```

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

__enumext_safe_exec:

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

```
\cs_new_protected:Nn \__enumext_safe_exec:
  {
    \__enumext_internal_mini_page:
    \__enumext_is_not_nested:
    \int_incr:N \l__enumext_level_int
    \int_compare:nNnT { \l__enumext_level_int } > { 4 }
      { \msg_fatal:nn { enumext } { list-too-deep } }
    \bool_set_true:N \l__enumext_standar_bool
    \__enumext_is_on_first_level:
  }
```

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

__enumext_parse_keys:n

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

```
\cs_new_protected:Npn \__enumext_parse_keys:n #1
  {
    \tl_if_novalue:nF {#1}
      {
        \str_clear:N \l__enumext_series_str
        \int_compare:nNnTF { \l__enumext_level_int } = { 1 }
          {
            \keys_set:nn { enumext / level-1 } {#1}
            \__enumext_parse_series:n {#1}
            \__enumext_nested_base_line_fix:
          }
          {
            \exp_args:Ne \keys_set:nn
              { enumext / level-\int_use:N \l__enumext_level_int } {#1}
          }
        \__enumext_store_active_keys:n {#1}
      }
  }
```

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

__enumext_start_store_level:
__enumext_stop_store_level:

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

```
\cs_new_protected:Nn \__enumext_start_store_level:
  {
    \bool_lazy_all:nT
      {
        { \bool_if_p:N \l__enumext_store_active_bool }
        { \bool_not_p:n { \l__enumext_keyans_env_bool } }
        { \bool_if_p:N \g__enumext_standar_bool }
      }
      {
        \int_compare:nNnT { \l__enumext_level_int } > { 1 }
          {
            \bool_set_true:c { l__enumext_store_upper_level_ \__enumext_level: _bool }
            \__enumext_store_level_open:
          }
      }
```

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

```
    \bool_lazy_all:nT
      {
        { \bool_if_p:N \l__enumext_store_active_bool }
        { \bool_not_p:n { \l__enumext_keyans_env_bool } }
        { \int_compare_p:nNn { \l__enumext_level_h_int } = { 1 } }
      }
      {
        \int_compare:nNnT { \l__enumext_level_int } > { 0 }
          {
            \bool_set_true:c { l__enumext_store_upper_level_ \__enumext_level: _bool }
            \__enumext_store_level_open:
          }
      }
  }
\cs_new_protected:Nn \__enumext_stop_store_level:
  {
    \bool_if:cT { l__enumext_store_upper_level_ \__enumext_level: _bool }
      {
        \__enumext_store_level_close:
      }
  }
```

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

__enumext_before_list: The function __enumext_before_list: will add the vertical spacing on the environment if the above key is active next to the {⟨*code*⟩} defined by the before* key if it is active.

```
\cs_new_protected:Nn \__enumext_before_list:
  {
    \__enumext_vspace_above:
    \__enumext_before_args_exec:
```

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

```
    \__enumext_check_ans_active:
```

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

```
    \dim_compare:nNnT
      { \dim_use:c { l__enumext_minipage_right_ \__enumext_level: _dim } } > { \c_zero_dim }
      {
        \dim_set:cn { l__enumext_minipage_left_ \__enumext_level: _dim }
          {
            \linewidth
            - \dim_use:c { l__enumext_minipage_right_ \__enumext_level: _dim }
            - \dim_use:c { l__enumext_minipage_hsep_ \__enumext_level: _dim }
          }
```

The boolean variable \l__enumext_minipage_active_X_bool will be activated and the integer variable \g__enumext_minipage_stat_int used by the \miniright command will be incremented, then the function __enumext_mini_addvspace: is called and the __enumext_mini_env* environment on the *"left side"* will be initialized followed by the *"vertical spacing"* applied to preserve the *"baseline"* between the *left* and *right* side environments. After these actions, the function __enumext_multicols_start: is called to handle the multicols environment.

Here we use the plain TEX macro \nointerlineskip to prevent baseline *"glue"* being added between the next pair of boxes in a *vertical list.*

```
        \bool_set_true:c { l__enumext_minipage_active_ \__enumext_level: _bool }
        \int_gincr:N \g__enumext_minipage_stat_int
        \__enumext_mini_addvspace:
        \nointerlineskip\noindent
        \begin{__enumext_mini_env*}
          { \dim_use:c { l__enumext_minipage_left_ \__enumext_level: _dim } }
      }
    \__enumext_multicols_start:
  }
```

(*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.

```
\cs_new_protected:Nn \__enumext_multicols_start:
  {
    \int_compare:nNnT
      { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } > { 1 }
      {
        \dim_compare:nNnT
          { \dim_use:c { l__enumext_columns_sep_ \__enumext_level: _dim } } = { \c_zero_dim }
          {
            \dim_set:cn { l__enumext_columns_sep_ \__enumext_level: _dim }
              {
                ( \dim_use:c { l__enumext_labelwidth_ \__enumext_level: _dim }
                  + \dim_use:c { l__enumext_labelsep_ \__enumext_level: _dim }
                ) / \int_use:c { l__enumext_columns_ \__enumext_level: _int }
                - \dim_use:c { l__enumext_listoffset_ \__enumext_level: _dim }
              }
          }
        \dim_set_eq:Nc \columnsep { l__enumext_columns_sep_ \__enumext_level: _dim  }
        \skip_zero:N \multicolsep
        \int_compare:nNnT { \l__enumext_level_int } > { 1 }
          {
            \dim_zero:N \columnseprule
          }
```

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.

```
        \bool_if:cF { l__enumext_minipage_active_ \__enumext_level: _bool }
          {
            \__enumext_multi_addvspace:
          }
        \raggedcolumns
        \begin{multicols}{ \int_use:c { l__enumext_columns_ \__enumext_level: _int } }
      }
  }
```

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

`\__enumext_multicols_stop:` The function `\__enumext_multicols_stop:` will stop the `multicols` environment. If the boolean variable `\l__enumext_minipage_active_X_bool` is false (not nested in `__enumext_mini_env*`) we will apply our *"vertical adjust"* spacing.

```
\cs_new_protected:Nn \__enumext_multicols_stop:
  {
    \int_compare:nNnT
      { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } > { 1 }
      {
        \end{multicols}
        \bool_if:cF { l__enumext_minipage_active_ \__enumext_level: _bool }
          {
            \par\addvspace{ \skip_use:c { l__enumext_multicols_below_ \__enumext_level: _skip } }
          }
      }
  }
```

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

`\__enumext_after_list:` The function `\__enumext_after_list:` will will check the state of the boolean variable `\l__enumext_minipage_active_X_bool`, if it is "true" a small test will be executed to check if we have omitted the use of `\miniright` (the `__enumext_mini_env*` environment has not been closed), then close `__enumext_mini_env*` and add the *adjusted vertical space* `\l__enumext_minipage_after_skip`, otherwise we will close the `multicols` environment.

```
\cs_new_protected:Nn \__enumext_after_list:
  {
    \bool_if:cTF { l__enumext_minipage_active_ \__enumext_level: _bool }
      {
        \int_compare:nNnT { \g__enumext_minipage_stat_int } = { 1 }
          {
            \msg_warning:nn { enumext } { missing-miniright }
            \miniright
          }
        \int_gzero:N \g__enumext_minipage_stat_int
        \end{__enumext_mini_env*}
        \par\addvspace { \l__enumext_minipage_after_skip }
      }
      { \__enumext_multicols_stop: }
```

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

```
     \__enumext_check_ans_key_hook:
```

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

```
    \__enumext_after_stop_list:
    \__enumext_vspace_below:
    \bool_set_false:N \l__enumext_standar_bool
    \__enumext_resume_save_counter:
  }
```

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

As we don't want our check to be executed `check-ans` by levels but on the complete list, we will take it out of the `enumext` environment using the *"hook"* function `\__enumext_after_env:nn`.

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

11.39 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.

```
\NewDocumentEnvironment{keyans}{ O{} }
  {
    \__enumext_keyans_safe_exec:
    \__enumext_keyans_parse_keys:n {#1}
    \__enumext_before_list_v:
    \__enumext_start_list:nn
      { \tl_use:N \l__enumext_label_v_tl }
      {
        \__enumext_list_arg_two_v:
        \__enumext_before_keys_exec_v:
      }
    \__enumext_after_args_exec_v:
  }
  {
    \__enumext_check_starred_cmd:n { item }
    \__enumext_stop_list:
    \__enumext_after_list_v:
  }
```

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

`\__enumext_keyans_safe_exec:` The `keyans` environment will only be available if the `save-ans` key is active and can only be used at the first level within the `enumext` environment. We do not want the environment to be nested, so we will set a maximum at this point. If the conditions are not met, an error message will be returned.

```
\cs_new_protected:Nn \__enumext_keyans_safe_exec:
  {
    \bool_if:NF \l__enumext_store_active_bool
      {
        \msg_error:nnnn { enumext } { wrong-place }{ keyans }{ save-ans }
      }
    \int_incr:N \l__enumext_keyans_level_int
    \bool_set_true:N \l__enumext_keyans_env_bool
    \__enumext_keyans_start_line:
    % Set false for interfering with enumext nested in keyans (yes, its possible and crayze)
    \bool_set_false:N \l__enumext_store_active_bool
    \int_compare:nNnT { \l__enumext_keyans_level_int } > { 1 }
      {
        \msg_error:nn { enumext } { keyans-nested }
      }
    \int_compare:nNnT { \l__enumext_level_int } > { 1 }
      {
        \msg_error:nn { enumext } { keyans-wrong-level }
      }
  }
```

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

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

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

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

`\__enumext_before_list_v:` The function `\__enumext_before_list_v:` will add the *vertical spacing above* the environment if the above key is active next to the ⟨*code*⟩ defined by the before key if it is active.

```
\cs_new_protected:Nn \__enumext_before_list_v:
  {
    \__enumext_vspace_above_v:
    \__enumext_before_args_exec_v:
```

When the mini-env key is active it will set the value of the `\l__enumext_minipage_right_v_dim` to be the *width* of the `__enumext_mini_env*` environment on the *left side*, using this value together with the value of the `\l__enumext_minipage_hsep_v_dim` set by the mini-sep key, the value of `\l__enumext_minipage_left_v_dim` will be set, which will be the *width* of `__enumextt_mini_env*` environment on the *right side*, always having `\linewidth` as the maximum width between them.

```
    \dim_compare:nNnT { \l__enumext_minipage_right_v_dim } > { \c_zero_dim }
      {
        \dim_set:Nn \l__enumext_minipage_left_v_dim
          {
            \linewidth - \l__enumext_minipage_right_v_dim - \l__enumext_minipage_hsep_v_dim
          }
```

The boolean variable `\l__enumext_minipage_active_v_bool` will be activated and the integer variable `\g__enumext_minipage_stat_int` used by the `\miniright` command will be incremented, then the function `\__enumext_keyans_mini_addvspace:` is called and the `__enumext_mini_env*` environment on *left side* will be initialized followed by the *vertical spacing* `\l__enumext_minipage_left_skip`. Here we use the plain TEX macro `\nointerlineskip` to prevent baseline *"glue"* being added between the next pair of boxes in a *vertical list*.

```
        \bool_set_true:N \l__enumext_minipage_active_v_bool
        \int_gincr:N \g__enumext_minipage_stat_int
        \__enumext_keyans_mini_addvspace:
        \nointerlineskip\noindent
        \begin{__enumext_mini_env*}{ \l__enumext_minipage_left_v_dim }
      }
```

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

```
    \__enumext_keyans_multicols_start:
  }
```

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

`\__enumext_keyans_multicols_start:` The function `\__enumext_keyans_multicols_start:` will start the multicols environment according to the value passed by the columns key.

```
\cs_new_protected:Nn \__enumext_keyans_multicols_start:
  {
    \int_compare:nNnT { \l__enumext_columns_v_int } > { 1 }
      {
```

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

```
        \dim_compare:nNnT { \l__enumext_columns_sep_v_dim } = { \c_zero_dim }
          {
            \dim_set:Nn \l__enumext_columns_sep_v_dim
              {
                (
                  \l__enumext_labelwidth_v_dim + \l__enumext_labelsep_v_dim
                ) / \l__enumext_columns_v_int
               - \l__enumext_listoffset_v_dim
              }
          }
        \dim_set_eq:NN \columnsep \l__enumext_columns_sep_v_dim
```

Then we will set the value of \multicolsep and \columnseprule equal to zero (we do not want a vertical rule in this environment).

```
        \skip_zero:N \multicolsep
        \dim_zero:N \columnseprule
```

We will calculate the *vertical spacing* settings for the multicols environment using the function __enumext_keyans_multi_addvspace: and apply our *"vertical adjust spacing"*, then start the multicols environment.

```
        \bool_if:NF \l__enumext_minipage_active_v_bool
          {
            \__enumext_keyans_multi_addvspace:
          }
        \raggedcolumns
        \begin{multicols}{ \l__enumext_columns_v_int }
      }
  }
```

(*End of definition for* __enumext_keyans_multicols_start:.)

__enumext_keyans_multicols_stop:

The function __enumext_keyans_multicols_stop: will stop the multicols environment. If the boolean variable \l__enumext_minipage_active_v_bool is false (not nested in __enumext_mini_-env*) we will apply our vertical "adjust" spacing.

```
\cs_new_protected:Nn \__enumext_keyans_multicols_stop:
  {
    \int_compare:nNnT { \l__enumext_columns_v_int } > { 1 }
      {
        \end{multicols}
        \bool_if:NF \l__enumext_minipage_active_v_bool
          {
            \par\addvspace{ \l__enumext_multicols_below_v_skip }
          }
      }
  }
```

(*End of definition for* __enumext_keyans_multicols_stop:.)

__enumext_after_list_v:

The function __enumext_after_list_v: will will check the state of the boolean variable \l__-enumext_minipage_active_v_bool, if it is "true" a small test will be executed to check if we have omitted the use of \miniright (the __enumext_mini_env* environment has not been closed), then close __enumext_mini_env* and add the vertical adjustment space \l__enumext_minipage_after_-skip, otherwise we will close the multicols environment.

```
\cs_new_protected:Nn \__enumext_after_list_v:
  {
    \bool_if:NTF \l__enumext_minipage_active_v_bool
      {
        \int_compare:nNnT { \g__enumext_minipage_stat_int } = { 1 }
          {
            \msg_warning:nn { enumext } { missing-miniright }
            \miniright
          }
        \int_gzero:N \g__enumext_minipage_stat_int
        \end{__enumext_mini_env*}
        \par\addvspace{ \l__enumext_minipage_after_skip }
      }
      { \__enumext_keyans_multicols_stop: }
```

Finally we will apply the {⟨*code*⟩} handled by the `after` key together with the *vertical space* handled by the `below` key if they are present.

```
    \bool_set_false:N \l__enumext_keyans_env_bool
    \__enumext_after_stop_list_v:
    \__enumext_vspace_below_v:
  }
```

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

11.40 The environment keyanspic and \anspic

The `keyanspic` environment is a list-based environment that uses the same configuration for *"spacing"* and ⟨*label*⟩ as the `keyans` environment, but it does not use `\item`.
The contents are passed to the environment by means of the `\anspic` command and are placed inside `minipage` environments, with the ⟨*label*⟩ underneath, adjusting widths according to the options passed to the environment.
Again it is necessary to "adjust" the spacing, both vertical and horizontal, to obtain an output like the one shown in the figure 12.

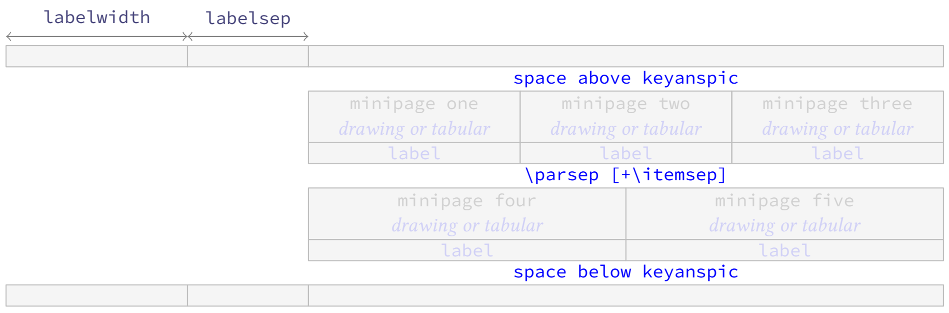


Figure 12: Representation of the keyanspic spacing in enumext.

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

11.40.1 The command \anspic

`\anspic` The `\anspic` command take three arguments, the starred (`*`) versions `\anspic*` and `\anspic*[`⟨*content*⟩`]` store the current ⟨*label*⟩ next to the `[`⟨*content*⟩`]` if it is present in the ⟨*sequence*⟩ and ⟨*prop list*⟩ defined by `save-ans` key. This command is used as a replacement for `\item` in the `keyanspic` environment.

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

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

```
    \bool_if:NF \l__enumext_store_active_bool
      {
        \msg_error:nnnn { enumext } { wrong-place }{ keyanspic }{ save-ans }
      }
    \int_compare:nNnT { \l__enumext_level_int } > { 1 }
      {
        \msg_error:nn { enumext } { keyanspic-wrong-level }
      }
    \int_compare:nNnT { \l__enumext_keyans_level_int } = { 1 }
      {
        \msg_error:nnnn { enumext } { command-wrong-place }{ anspic }{ keyans }
      }
```

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.

```
    \seq_put_right:Nn \l__enumext_keyans_pic_body_seq
      {
        \__enumext_keyans_anspic_code:nnn { #1 } { #2 } { #3 }
      }
  }
```

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

`\__enumext_keyans_anspic_code:nnn` The function `\__enumext_keyans_anspic_code:nnn` will be in charge of handling the *"counter"* and ⟨*label*⟩, which will have the same configuration as the `keyans` environment.

```
\cs_new_protected:Nn \__enumext_keyans_anspic_code:nnn
  {
    \stepcounter { enumXvi }
    #3 \\
    \bool_if:nT { #1 }
      {
        \__enumext_keyans_addto_prop:n { #2 }
        \__enumext_keyans_store_ref:
        \__enumext_keyans_addto_seq:n { #2 }
        \int_gincr:N \g__enumext_check_starred_cmd_int
        \bool_lazy_or:nnT
          { \bool_if_p:N \l__enumext_show_answer_bool }
          { \bool_if_p:N \l__enumext_show_position_bool }
          {
            \tl_set_eq:NN \l__enumext_label_v_tl \l__enumext_label_vi_tl
            \__enumext_keyans_show_left:n { #2 }
            \tl_set_eq:NN \l__enumext_label_vi_tl \l__enumext_label_v_tl
          }
      }
    \tl_use:N \l__enumext_label_font_style_v_tl
    \__enumext_wrapper_label_v:n { \l__enumext_label_vi_tl } \__enumext_keyans_show_item_opt:
  }
```

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

11.40.2 The environment keyanspic

keyanspic Now we define the environment keyanspic based on list. The optional argument [⟨*number above, number below*⟩] will determine the number of minipage environments that will be above and below separated by \parsep+\itemsep within it.

```
\NewDocumentEnvironment{keyanspic}{ o }
  {
    \__enumext_keyans_pic_safe_exec:
    \__enumext_start_list:nn
      { }
      {
        \__enumext_keyans_pic_arg_two:
      }
```

We apply the "adjusted" vertical spacing above the environment

```
    \vspace { \l__enumext_keyans_pic_above_skip }
  }
```

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.

```
  {
    \tl_if_novalue:nTF { #1 }
      {
        \__enumext_keyans_pic_do:e { \seq_count:N \l__enumext_keyans_pic_body_seq }
      }
      { \__enumext_keyans_pic_do:n { #1 } }
    \__enumext_stop_list:
    \__enumext_check_starred_cmd:n { anspic }
    \setcounter { enumXvi } { 0 }
    \vspace { \l__enumext_topsep_v_skip }
    %\bool_set_false:N \l__enumext_store_active_bool
  }
```

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

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

```
\cs_new_protected:Nn \__enumext_keyans_pic_safe_exec:
  {
    \int_incr:N \l__enumext_keyans_pic_level_int
    \int_compare:nNnT { \l__enumext_keyans_pic_level_int } > { 1 }
      {
        \msg_error:nn { enumext } { keyanspic-nested }
      }
    \__enumext_keyans_start_line:
  }
```

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

__enumext_keyans_pic_skip_abs:N

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

```
\cs_new_protected:Npn \__enumext_keyans_pic_skip_abs:N #1
  {
    \dim_compare:nNnT { #1 } < { 0pt }
      { \skip_set:Nn #1 { -#1 } }
  }
```

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

__enumext_keyans_pic_arg_two:

The function `\__enumext_keyans_pic_arg_two:` will be used in the second argument of the `\__enumext_start_list:nn` function that defines the `keyanspic` environment, it will handle the setting of spaces.

```
\cs_new_protected:Nn \__enumext_keyans_pic_arg_two:
  {
```

The first thing to do is to set the boolean variable `\l__enumext_leftmargin_tmp_v_bool` handled by the `list-indent` key to false, then we copy the definition of the second list argument from the `keyans` environment.

```
    \bool_set_false:N \l__enumext_leftmargin_tmp_v_bool
    \__enumext_list_arg_two_v:
```

We will add the value of `\itemsep` to `\parsep` which we will use as vertical spacing between the above and below `minipage` environments. and adjust the value of `\leftmargin`, the label and counter are handled directly by the `\anspic` command. Then we make equal to zero `\labelwidth`, `\labelsep`, `\partopsep` and `\itemsep` so that the horizontal and vertical spacing is not affected.

```
    \skip_add:Nn \parsep { \itemsep }
    \dim_add:Nn  \leftmargin { -\labelwidth - \labelsep }
    \dim_zero:N  \labelwidth
    \dim_zero:N  \listparindent
    \dim_zero:N  \labelsep
    \skip_zero:N \partopsep
    \skip_zero:N \itemsep
```

We set the value of `\l__enumext_keyans_pic_above_skip` which we will use to apply our "adjust" space above `keyanspic`, finally we call `\__enumext_item_std:w` followed by `\scan_stop:` to prevent the error message returned by LaTeX when not using the `\item` command.

```
    \__enumext_keyans_pic_skip_abs:N \parsep
    \skip_set:Nn \l__enumext_keyans_pic_above_skip
      {
        \box_dp:N \strutbox
        + \l__enumext_topsep_v_skip
        - \parsep
      }
    \__enumext_item_std:w \scan_stop:
  }
```

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

__enumext_keyans_pic_do:n
__enumext_keyans_pic_do:e

The optional argument is split by comma and is handled directly by the function `\__enumext_keyans_pic_do:n` and passed to the function `\__enumext_keyans_pic_row:n`.

```
\cs_new_protected:Nn \__enumext_keyans_pic_do:n
  {
    \clist_map_function:nN { #1 } \__enumext_keyans_pic_row:n
  }
\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.

```
\cs_new_protected:Nn \__enumext_keyans_pic_row:n
  {
    \dim_set:Nn \l__enumext_keyans_pic_width_dim { \linewidth / #1 }
    \int_set:Nn \l__enumext_keyans_pic_above_int { \l__enumext_keyans_pic_below_int }
    \int_set:Nn \l__enumext_keyans_pic_below_int { \l__enumext_keyans_pic_above_int + #1 }
    \int_step_inline:nnn
      { \l__enumext_keyans_pic_above_int + 1 }
      { \l__enumext_keyans_pic_below_int }
      {
        \__enumext_minipage:w [ b ]{ \l__enumext_keyans_pic_width_dim }
          \centering
          \seq_item:Nn \l__enumext_keyans_pic_body_seq { ##1 }
        \__enumext_endminipage:
      }
    \par
  }
```

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

11.41 The horizontal environments

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

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

11.41.1 Functions for item box width

`\__enumext_starred_columns_set_vii:`
`\__enumext_starred_columns_set_viii:`

We set the default value for the *width of the box* containing the content of the items for `enumext*` environment.

```
\cs_new_protected:Nn \__enumext_starred_columns_set_vii:
  {
    \dim_compare:nNnT { \l__enumext_columns_sep_vii_dim } = { \c_zero_dim }
      {
        \dim_set:Nn \l__enumext_columns_sep_vii_dim
          {
            ( \l__enumext_labelwidth_vii_dim + \l__enumext_labelsep_vii_dim )
            / \l__enumext_columns_vii_int
          }
      }
    \int_set:Nn \l__enumext_tmpa_vii_int { \l__enumext_columns_vii_int - 1 }
    \dim_set:Nn \l__enumext_item_width_vii_dim
      {
        ( \linewidth - \l__enumext_columns_sep_vii_dim * \l__enumext_tmpa_vii_int )
        / \l__enumext_columns_vii_int
        - \l__enumext_labelwidth_vii_dim
        - \l__enumext_labelsep_vii_dim
      }
```

When the key `rightmargin` is active we must adjust the values.

```
    \dim_compare:nNnT { \l__enumext_rightmargin_vii_dim } > { \c_zero_dim }
      {
        \dim_sub:Nn \l__enumext_item_width_vii_dim
          {
            ( \l__enumext_rightmargin_vii_dim * \l__enumext_tmpa_vii_int )
            / \l__enumext_columns_vii_int
          }
        \dim_add:Nn \l__enumext_columns_sep_vii_dim
          {
            \l__enumext_rightmargin_vii_dim
          }
      }
  }
```

Same implementation for the `keyans*` environment.

```
\cs_new_protected:Nn \__enumext_starred_columns_set_viii:
  {
    \dim_compare:nNnT { \l__enumext_columns_sep_viii_dim } = { \c_zero_dim }
      {
        \dim_set:Nn \l__enumext_columns_sep_viii_dim
          {
            ( \l__enumext_labelwidth_viii_dim + \l__enumext_labelsep_viii_dim )
            / \l__enumext_columns_viii_int
          }
      }
    \int_set:Nn \l__enumext_tmpa_viii_int { \l__enumext_columns_viii_int - 1 }
    \dim_set:Nn \l__enumext_item_width_viii_dim
      {
        ( \linewidth - \l__enumext_columns_sep_viii_dim * \l__enumext_tmpa_viii_int )
        / \l__enumext_columns_viii_int
        - \l__enumext_labelwidth_viii_dim
        - \l__enumext_labelsep_viii_dim
      }
    \dim_compare:nNnT { \l__enumext_rightmargin_viii_dim } > { \c_zero_dim }
      {
        \dim_sub:Nn \l__enumext_item_width_viii_dim
          {
            ( \l__enumext_rightmargin_viii_dim * \l__enumext_tmpa_vii_int )
            / \l__enumext_columns_viii_int
          }
        \dim_add:Nn \l__enumext_columns_sep_viii_dim
          {
            \l__enumext_rightmargin_viii_dim
          }
      }
  }
```

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

11.41.2 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 `\itemwidthdim` for the `enumext*` environment.

```
\cs_new_protected:Npn \__enumext_starred_joined_item_vii:n #1
  {
    \int_set:Nn \l__enumext_joined_item_vii_int {#1}
    \int_compare:nNnT { \l__enumext_joined_item_vii_int } > { \l__enumext_columns_vii_int }
      {
        \msg_warning:nnee { enumext } { item-joined }
          { \int_use:N \l__enumext_joined_item_vii_int }
          { \int_use:N \l__enumext_columns_vii_int }
        \int_set:Nn \l__enumext_joined_item_vii_int
          {
            \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + 1
          }
      }
    \int_compare:nNnT
      { \l__enumext_joined_item_vii_int }
        >
      { \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + 1 }
      {
        \msg_warning:nnee { enumext } { item-joined-columns }
          { \int_use:N \l__enumext_joined_item_vii_int }
          {
            \int_eval:n
              { \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + 1 }
          }
        \int_set:Nn \l__enumext_joined_item_vii_int
          {
            \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + 1
          }
      }
    \int_compare:nNnTF { \l__enumext_joined_item_vii_int } > { 1 }
      {
        \int_set_eq:NN \l__enumext_joined_item_aux_vii_int \l__enumext_joined_item_vii_int
        \int_decr:N \l__enumext_joined_item_aux_vii_int
        \int_add:Nn \l__enumext_item_column_pos_vii_int { \l__enumext_joined_item_aux_vii_int }
        \int_gadd:Nn \g__enumext_item_count_all_vii_int { \l__enumext_joined_item_aux_vii_int }
        \dim_set:Nn \l__enumext_joined_width_vii_dim
          {
            \l__enumext_item_width_vii_dim * \l__enumext_joined_item_vii_int
            + (  \l__enumext_labelwidth_vii_dim + \l__enumext_labelsep_vii_dim
               + \l__enumext_columns_sep_vii_dim
              )*\l__enumext_joined_item_aux_vii_int
          }
        \dim_set_eq:NN \itemwidthdim \l__enumext_joined_width_vii_dim
      }
      {
        \dim_set_eq:NN \l__enumext_joined_width_vii_dim \l__enumext_item_width_vii_dim
        \dim_set_eq:NN \itemwidthdim \l__enumext_item_width_vii_dim
      }
  }
```

Same implementation for the keyans* environment.

```
\cs_new_protected:Npn \__enumext_starred_joined_item_viii:n #1
  {
    \int_set:Nn \l__enumext_joined_item_viii_int {#1}
    \int_compare:nNnT { \l__enumext_joined_item_viii_int } > { \l__enumext_columns_viii_int }
      {
        \msg_warning:nnee { enumext } { item-joined }
          { \int_use:N \l__enumext_joined_item_viii_int }
          { \int_use:N \l__enumext_columns_viii_int }
        \int_set:Nn \l__enumext_joined_item_viii_int
          {
            \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1
          }
      }
    \int_compare:nNnT
      { \l__enumext_joined_item_viii_int }
        >
      { \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1 }
      {
        \msg_warning:nnee { enumext } { item-joined-columns }
          { \int_use:N \l__enumext_joined_item_viii_int }
          {
            \int_eval:n
              { \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1 }
          }
        \int_set:Nn \l__enumext_joined_item_viii_int
          {
            \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1
          }
      }
    \int_compare:nNnTF { \l__enumext_joined_item_viii_int } > { 1 }
      {
        \int_set_eq:NN \l__enumext_joined_item_aux_viii_int \l__enumext_joined_item_viii_int
        \int_decr:N \l__enumext_joined_item_aux_viii_int
        \int_add:Nn \l__enumext_item_column_pos_viii_int { \l__enumext_joined_item_aux_viii_int }
        \int_gadd:Nn \g__enumext_item_count_all_viii_int { \l__enumext_joined_item_aux_viii_int }
        \dim_set:Nn \l__enumext_joined_width_viii_dim
          {
            \l__enumext_item_width_viii_dim * \l__enumext_joined_item_viii_int
            + (  \l__enumext_labelwidth_viii_dim + \l__enumext_labelsep_viii_dim
               + \l__enumext_columns_sep_viii_dim
              )*\l__enumext_joined_item_aux_viii_int
          }
        \dim_set_eq:NN \itemwidthdim \l__enumext_joined_width_viii_dim
      }
      {
        \dim_set_eq:NN \l__enumext_joined_width_viii_dim \l__enumext_item_width_viii_dim
        \dim_set_eq:NN \itemwidthdim \l__enumext_item_width_viii_dim
      }
  }
```

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

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

__enumext_start_mini_vii:
__enumext_stop_mini_vii:

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

```
\cs_new_protected:Nn \__enumext_start_mini_vii:
  {
    \dim_compare:nNnT { \l__enumext_minipage_right_vii_dim } > { \c_zero_dim }
      {
        \dim_set:Nn \l__enumext_minipage_left_vii_dim
          {
            \linewidth
            - \l__enumext_minipage_right_vii_dim
            - \l__enumext_minipage_hsep_vii_dim
          }
        \bool_set_true:N \l__enumext_minipage_active_vii_bool
        \dim_gset_eq:NN
          \g__enumext_minipage_right_vii_dim
          \l__enumext_minipage_right_vii_dim
        \__enumext_mini_addvspace_vii:
        \nointerlineskip\noindent
        \begin{__enumext_mini_env*}{ \l__enumext_minipage_left_vii_dim }
      }
    }
```

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

```
\cs_new_protected:Nn \__enumext_stop_mini_vii:
  {
    \bool_if:NT \l__enumext_minipage_active_vii_bool
      {
        \end{__enumext_mini_env*}
        \hfill
        \bool_gset_true:N \g__enumext_minipage_active_vii_bool
      }
  }
```

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

```
\__enumext_after_env:nn {enumext*}
  {
    \bool_if:NT \g__enumext_minipage_active_vii_bool
      {
        \begin{__enumext_mini_env*}{ \g__enumext_minipage_right_vii_dim }
          \par\addvspace { \g__enumext_minipage_right_skip }
          \bool_if:NF \g__enumext_minipage_center_vii_bool
            {
              \tl_put_left:Nn \g__enumext_miniright_code_vii_tl
                {
                  \centering
                }
            }
          \vbox_set_top:Nn \l__enumext_miniright_code_vii_box
            {
              \tl_use:N \g__enumext_miniright_code_vii_tl
            }
          \box_use_drop:N \l__enumext_miniright_code_vii_box
        \end{__enumext_mini_env*}
        \par\addvspace{ \g__enumext_minipage_after_skip }
      }
    \bool_gset_false:N \g__enumext_minipage_active_vii_bool
    \bool_gset_true:N \g__enumext_minipage_center_vii_bool
    \tl_gclear:N \g__enumext_miniright_code_vii_tl
    \dim_gzero:N \g__enumext_minipage_right_vii_dim
    \bool_gset_false:N \g__enumext_starred_bool
  }
```

(*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.

```
\cs_new_protected:Nn \__enumext_start_mini_viii:
  {
    \dim_compare:nNnT { \l__enumext_minipage_right_viii_dim } > { \c_zero_dim }
      {
        \dim_set:Nn \l__enumext_minipage_left_viii_dim
          {
            \linewidth
            - \l__enumext_minipage_right_viii_dim
            - \l__enumext_minipage_hsep_viii_dim
          }
        \bool_set_true:N \l__enumext_minipage_active_viii_bool
        \dim_gset_eq:NN
          \g__enumext_minipage_right_viii_dim
          \l__enumext_minipage_right_viii_dim
        \__enumext_mini_addvspace_viii:
        \nointerlineskip\noindent
        \begin{__enumext_mini_env*}{ \l__enumext_minipage_left_viii_dim }
      }
   }
\cs_new_protected:Nn \__enumext_stop_mini_viii:
  {
    \bool_if:NT \l__enumext_minipage_active_viii_bool
      {
        \end{__enumext_mini_env*}
        \hfill
        \bool_gset_true:N \g__enumext_minipage_active_viii_bool
      }
   }
\__enumext_after_env:nn {keyans*}
  {
    \bool_if:NT \g__enumext_minipage_active_viii_bool
      {
        \begin{__enumext_mini_env*}{ \g__enumext_minipage_right_viii_dim }
          \par\addvspace { \g__enumext_minipage_right_skip }
          \bool_if:NF \g__enumext_minipage_center_viii_bool
            {
              \tl_put_left:Nn \g__enumext_miniright_code_viii_tl
                {
                  \centering
                }
            }
          \vbox_set_top:Nn \l__enumext_miniright_code_viii_box
            {
              \tl_use:N \g__enumext_miniright_code_viii_tl
            }
          \box_use_drop:N \l__enumext_miniright_code_viii_box
        \end{__enumext_mini_env*}
        \par\addvspace{ \g__enumext_minipage_after_skip }
      }
    \bool_gset_false:N \g__enumext_minipage_active_viii_bool
    \bool_gset_true:N \g__enumext_minipage_center_viii_bool
    \tl_gclear:N \g__enumext_miniright_code_viii_tl
    \dim_gzero:N \g__enumext_minipage_right_viii_dim
  }
```

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

11.42 The environment enumext*

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

```
\NewDocumentEnvironment{enumext*}{ o }
  {
    \__enumext_safe_exec_vii:
    \__enumext_parse_keys_vii:n {#1}
    \__enumext_before_list_vii:
    \__enumext_start_store_level_vii:
    \__enumext_start_list:nn { }
      {
        \__enumext_list_arg_two_vii:
        \__enumext_before_keys_exec_vii:
      }
    \__enumext_starred_columns_set_vii:
    \item[] \scan_stop:
    \cs_set_eq:NN \__enumext_stop_item_tmp_vii: \noindent
    \cs_set_eq:NN \item \__enumext_start_item_tmp_vii:
  }
  {
    \__enumext_stop_item_tmp_vii:
    \__enumext_remove_extra_parsep_vii:
    \__enumext_stop_list:
```

```
    \__enumext_stop_store_level_vii:
    \__enumext_after_list_vii:
  }
```

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

`\__enumext_safe_exec_vii:` We will first call the function `\__enumext_internal_mini_page:` to create the environment `__enumext_mini_env*`, 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.

```
\cs_new_protected:Nn \__enumext_safe_exec_vii:
  {
    \__enumext_internal_mini_page:
    \__enumext_is_not_nested:
    \int_incr:N \l__enumext_level_h_int
    \int_compare:nNnT { \l__enumext_level_h_int } > { 1 }
      {
        \msg_error:nn { enumext } { nested }
      }
    \bool_set_true:N \l__enumext_starred_bool
    \__enumext_is_on_first_level:
  }
```

(*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`.

```
\cs_new_protected:Npn \__enumext_parse_keys_vii:n #1
  {
    \tl_if_novalue:nF {#1}
      {
        \str_clear:N \l__enumext_series_str
        \keys_set:nn { enumext / enumext* } {#1}
        \__enumext_parse_series:n {#1}
        \__enumext_store_active_keys_vii:n {#1}
        \__enumext_nested_base_line_fix:
      }
  }
```

(*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 function `\__enumext_start_mini_vii:` used by the keys `mini-env`, `mini-right` and `mini-right*`.

```
\cs_new_protected:Nn \__enumext_before_list_vii:
  {
    \__enumext_vspace_above_vii:
    \__enumext_check_ans_active:
    \__enumext_before_args_exec_vii:
    \__enumext_start_mini_vii:
  }
```

(*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.

```
\cs_new_protected:Nn \__enumext_after_list_vii:
  {
    \__enumext_stop_mini_vii:
    \__enumext_after_stop_list_vii:
```

```
    \__enumext_check_ans_key_hook:
    \__enumext_vspace_below_vii:
    \bool_set_false:N \l__enumext_starred_bool
    \__enumext_resume_save_counter:
  }
```

(*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`.

```
\cs_new_protected:Nn \__enumext_start_store_level_vii:
  {
    \bool_if:NT \l__enumext_store_active_bool
      {
        \int_compare:nNnT { \l__enumext_level_int } > { 0 }
          {
            \__enumext_store_level_open_vii:
          }
      }
  }
\cs_new_protected:Nn \__enumext_stop_store_level_vii:
  {
    \bool_if:NT \l__enumext_store_active_bool
      {
        \int_compare:nNnT { \l__enumext_level_int } > { 0 }
          {
            \__enumext_store_level_close_vii:
          }
      }
  }
```

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

11.42.1 The command `\item` in enumext*

`\__enumext_start_item_tmp_vii:`

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

```
\cs_new_protected_nopar:Nn \__enumext_start_item_tmp_vii:
  {
    \__enumext_stop_item_tmp_vii:
    \int_incr:N \l__enumext_item_column_pos_vii_int
    \int_gincr:N \g__enumext_item_count_all_vii_int
    \__enumext_item_peek_args_vii:
  }
```

(*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)`.

```
\cs_new_protected:Nn \__enumext_item_peek_args_vii:
  {
    \peek_meaning:NTF (
      { \__enumext_joined_item_vii:w }
      { \__enumext_joined_item_vii:w (1) }
  }
```

(*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`.

```
\cs_new_protected:Npn \__enumext_joined_item_vii:w (#1)
  {
    \__enumext_starred_joined_item_vii:n {#1}
    \peek_meaning_remove:NTF *
      { \__enumext_starred_item_vii:w  }
      { \__enumext_standar_item_vii:w }
  }
```

(*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].

```
\cs_new_protected:Npn \__enumext_standar_item_vii:w
  {
    \bool_set_false:N \l__enumext_item_starred_vii_bool
      \peek_meaning:NTF [
        {
          \bool_set_eq:NN
            \l__enumext_wrap_label_vii_bool
            \l__enumext_wrap_label_opt_vii_bool
          \__enumext_start_item_vii:w
        }
        {
          \bool_set_true:N \l__enumext_wrap_label_vii_bool
          \legacy_if_set_true:n { @noitemarg }
          \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
        }
  }
```

(*End of definition for* __enumext_standar_item_vii:w.)

__enumext_starred_item_vii:w
__enumext_starred_item_vii_aux_i:w
__enumext_starred_item_vii_aux_ii:w
__enumext_starred_item_vii_aux_iii:w

The function __enumext_starred_item_vii:w together with the specified auxiliary functions aux_i:w, aux_ii:w, and aux_iii:w execute \item*, \item*[⟨*symbol*⟩] and \item*[⟨*symbol*⟩][⟨*offset*⟩].

```
\cs_new_protected:Npn \__enumext_starred_item_vii:w
  {
    \bool_set_true:N \l__enumext_item_starred_vii_bool
    \bool_set_true:N \l__enumext_wrap_label_vii_bool
    \peek_meaning:NTF [
      { \__enumext_starred_item_vii_aux_i:w }
      { \__enumext_starred_item_vii_aux_ii:w }
  }
\cs_new_protected:Npn \__enumext_starred_item_vii_aux_i:w [#1]
  {
    \tl_gset:Nn \g__enumext_item_symbol_aux_vii_tl {#1}
    \__enumext_starred_item_vii_aux_ii:w
  }
\cs_new_protected:Npn \__enumext_starred_item_vii_aux_ii:w
  {
    \peek_meaning:NTF [
      { \__enumext_starred_item_vii_aux_iii:w }
      {
        \dim_set_eq:NN
          \l__enumext_item_symbol_sep_vii_dim
          \l__enumext_labelsep_vii_dim
        \legacy_if_set_true:n { @noitemarg }
        \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
      }
  }
\cs_new_protected:Npn \__enumext_starred_item_vii_aux_iii:w [#1]
  {
    \dim_set:Nn \l__enumext_item_symbol_sep_vii_dim {#1}
    \legacy_if_set_true:n { @noitemarg }
    \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
  }
```

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

11.42.2 Real definition of \item in enumext*

__enumext_start_item_vii:w

The functions __enumext_start_item_vii:w and __enumext_stop_item_vii: executing the true definition of \item inside the enumext* environment. The first thing we will do is set the value of __enumext_stop_item_tmp_vii: equal to __enumext_stop_item_vii: which we will define later and add the hyperref compatible enumXvii counter, after that we will start capturing the item content in a box. Here need setting the \if@hyper@item switch to *"true"* for hyperref compatible. The explanation for this is given by the master Heiko Oberdiek on \refstepcounter{enumi} twice (or more) creates destination with the same identifier.

```
\cs_new_protected_nopar:Npn \__enumext_start_item_vii:w [#1]
  {
    \cs_set_eq:NN \__enumext_stop_item_tmp_vii: \__enumext_stop_item_vii:
    \legacy_if:nT { @noitemarg }
      {
        \legacy_if_set_false:n { @noitemarg }
        \legacy_if:nT { @nmbrlist }
          {
            \bool_if:NT \l__enumext_hyperref_bool
              {
                \legacy_if_set_true:n { @hyper@item }
              }
            \refstepcounter{enumXvii}
            \bool_if:NT \l__enumext_check_answers_bool
              {
                \int_gincr:N \g__enumext_item_number_int
                \bool_set_true:N \l__enumext_item_number_bool
              }
          }
      }
```

Here we start capturing \item and its contents into a group using the plain form of the lrbox environment. If the state of the variable \l__enumext_footnotes_key_bool is false, we will redefine the command \footnote, followed by printing the ⟨*symbol*⟩ defined for \item* if it is present and open a new group inside which we execute font key next to \item and the keys wrap-label, wrap-label*, align, close the group and execute the key labelsep and then the key first. Finally we open the minipage environment and execute the listparindent key which will be equal to \parindent, the parsep key which will be equal to \parskip and the itemindent key.

```
    \group_begin:
      \lrbox{ \l__enumext_item_text_vii_box }
        \bool_if:NF \l__enumext_footnotes_key_bool
          {
            \__enumext_renew_footnote:
          }
        \bool_if:NT \l__enumext_item_starred_vii_bool
          {
            \tl_if_blank:VT \g__enumext_item_symbol_aux_vii_tl
              {
                \tl_gset_eq:NN
                  \g__enumext_item_symbol_aux_vii_tl \l__enumext_item_symbol_vii_tl
              }
            \mode_leave_vertical:
            \skip_horizontal:n { -\l__enumext_item_symbol_sep_vii_dim }
            \makebox[ 0pt ][ r ]{ \g__enumext_item_symbol_aux_vii_tl }
            \skip_horizontal:N \l__enumext_item_symbol_sep_vii_dim
            \tl_gclear:N \g__enumext_item_symbol_aux_vii_tl
          }
        \group_begin:
          \tl_use:N \l__enumext_label_font_style_vii_tl
          \bool_if:NTF \l__enumext_wrap_label_vii_bool
            {
              \makebox[ \l__enumext_labelwidth_vii_dim ][ \l__enumext_align_label_vii_str ]
                { \__enumext_wrapper_label_vii:n {#1} }
            }
            {
              \makebox[ \l__enumext_labelwidth_vii_dim ][ \l__enumext_align_label_vii_str ]{ #1 }
            }
        \group_end:
        \skip_horizontal:N \l__enumext_labelsep_vii_dim
        \tl_use:N \l__enumext_after_list_args_vii_tl
        \__enumext_minipage:w [ t ]{ \l__enumext_joined_width_vii_dim  }
          \skip_set_eq:NN \parindent \l__enumext_listparindent_vii_dim
          \skip_set_eq:NN \parskip \l__enumext_parsep_vii_skip
          \tl_use:N \l__enumext_fake_item_indent_vii_tl
    }
```

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

`\__enumext_stop_item_vii:` The function `\__enumext_stop_item_vii:` shall terminate with the capture of `\item` and its ⟨*contents*⟩. Close the environments `minipage`, `lrbox` and the group. Then we only have to set the width of the box and print it next to `\footnote`, and add the horizontal and vertical separation between the boxes.

```
\cs_new_protected_nopar:Nn \__enumext_stop_item_vii:
  {
        \__enumext_endminipage:
      \endlrbox
    \group_end:
    \box_set_wd:Nn \l__enumext_item_text_vii_box
      {
        \l__enumext_joined_width_vii_dim
        + \l__enumext_labelwidth_vii_dim
        + \l__enumext_labelsep_vii_dim
      }
    \int_set:Nn \hbadness { 10000 }
    \box_use_drop:N \l__enumext_item_text_vii_box
    \bool_if:NF \l__enumext_footnotes_key_bool
      {
        \__enumext_print_footnote:
      }
    \int_compare:nNnTF { \l__enumext_item_column_pos_vii_int } = { \l__enumext_columns_vii_int }
      {
        \par\noindent
        \int_zero:N \l__enumext_item_column_pos_vii_int
      }
      { \hspace{ \l__enumext_columns_sep_vii_dim } }
  }
```

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

`\__enumext_remove_extra_parsep_vii:` Finally we will remove the vertical space equal to `\parsep` when the total number of items is divisible by the number of items in the last row of the environment.

```
\cs_new_protected:Nn \__enumext_remove_extra_parsep_vii:
  {
    \int_compare:nNnT
      {
        \int_mod:nn {  \g__enumext_item_count_all_vii_int } { \l__enumext_columns_vii_int }
      }
      =
      { 0 }
      {
        \par
        \vspace{ -\l__enumext_itemsep_vii_skip }
        \int_gzero:N \g__enumext_item_count_all_vii_int
      }
  }
```

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`.

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

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

11.43 The environment keyans*

`keyans*` First we will generate the environment and we will give a temporary definition to `\__enumext_stop_item_tmp_viii:` equal to `\noindent` and next to `\item` equal to `\__enumext_start_item_tmp_viii:` which we will redefine later.

```
\NewDocumentEnvironment{keyans*}{ o }
  {
    \__enumext_safe_exec_viii:
    \__enumext_parse_keys_viii:n {#1}

    \__enumext_before_list_viii:
    \__enumext_start_list:nn { }
```

```
      {
        \__enumext_list_arg_two_viii:
        \__enumext_before_keys_exec_viii:
      }
    \__enumext_starred_columns_set_viii:
    \item[] \scan_stop:
    \cs_set_eq:NN \__enumext_stop_item_tmp_viii: \noindent
    \cs_set_eq:NN \item \__enumext_start_item_tmp_viii:
  }
  {
    \__enumext_stop_item_tmp_viii:
    \__enumext_remove_extra_parsep_viii:
    \__enumext_check_starred_cmd:n { item }
    \__enumext_stop_list:
    \__enumext_after_list_viii:
  }
```

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

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

```
\cs_new_protected:Nn \__enumext_safe_exec_viii:
  {
    \int_incr:N \l__enumext_keyans_level_h_int
    \int_compare:nNnT { \l__enumext_keyans_level_h_int } > { 1 }
      {
        \msg_error:nn { enumext } { nested }
      }
    \__enumext_keyans_start_line:
    % Set false for interfering with enumext nested in keyans* (yes, its possible and crayze)
    \bool_set_false:N \l__enumext_store_active_bool
    \int_compare:nNnT { \l__enumext_level_int } > { 1 }
      {
        \msg_error:nn { enumext } { keyans-wrong-level }
      }
  }
```

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

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

```
\cs_new_protected:Npn \__enumext_parse_keys_viii:n #1
  {
    \tl_if_novalue:nF {#1}
      {
        \keys_set:nn { enumext / keyans* } {#1}
      }
  }
```

(*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.

```
\cs_new_protected:Nn \__enumext_before_list_viii:
  {
    \__enumext_vspace_above_viii:
    \__enumext_before_args_exec_viii:
    \__enumext_start_mini_viii:
  }
```

(*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.

```
\cs_new_protected:Nn \__enumext_after_list_viii:
  {
    \__enumext_stop_mini_viii:
    \__enumext_after_stop_list_viii:
    \__enumext_vspace_below_viii:
  }
```

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

11.43.1 The command \item in keyans*

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

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

```
\cs_new_protected_nopar:Nn \__enumext_start_item_tmp_viii:
  {
    \__enumext_stop_item_tmp_viii:
    \int_incr:N \l__enumext_item_column_pos_viii_int
    \int_gincr:N \g__enumext_item_count_all_viii_int
    \__enumext_item_peek_args_viii:
  }
```

(*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).

```
\cs_new_protected:Nn \__enumext_item_peek_args_viii:
  {
    \peek_meaning:NTF (
      { \__enumext_joined_item_viii:w }
      { \__enumext_joined_item_viii:w (1) }
  }
```

(*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.

```
\cs_new_protected:Npn \__enumext_joined_item_viii:w (#1)
  {
    \__enumext_starred_joined_item_viii:n {#1}
    \peek_meaning_remove:NTF *
      { \__enumext_starred_item_viii:w  }
      { \__enumext_standar_item_viii:w }
  }
```

(*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].

```
\cs_new_protected:Npn \__enumext_standar_item_viii:w
  {
    \bool_set_false:N \l__enumext_item_starred_viii_bool
      \peek_meaning:NTF [
        {
          \bool_set_eq:NN
            \l__enumext_wrap_label_viii_bool
            \l__enumext_wrap_label_opt_viii_bool
          \__enumext_start_item_viii:w
        }
        {
          \bool_set_true:N \l__enumext_wrap_label_viii_bool
```

```
            \legacy_if_set_true:n { @noitemarg }
            \__enumext_start_item_viii:w [ \l__enumext_label_viii_tl ]
          }
    }
```

(*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⟩]`.

```
\cs_new_protected:Npn \__enumext_starred_item_viii:w
  {
    \bool_set_true:N \l__enumext_item_starred_viii_bool
    \bool_set_true:N \l__enumext_wrap_label_viii_bool
    \peek_meaning:NTF [
      { \__enumext_starred_item_viii_aux_i:w }
      { \__enumext_starred_item_viii_aux_ii:w }
  }
```

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`.

```
\cs_new_protected:Npn \__enumext_starred_item_viii_aux_i:w [#1]
  {
    \tl_clear:N \l__enumext_store_current_label_tl
    \tl_if_novalue:nF { #1 }
      {
        \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_tl
          {
            \tl_put_right:Ne \l__enumext_store_current_label_tl { \l__enumext_store_keyans_item_op
            \tl_put_right:Ne \l__enumext_store_current_label_tl { #1 }
          }
        \tl_set:Ne \l__enumext_store_current_opt_arg_tl { #1 }
      }
    \__enumext_starred_item_viii_aux_ii:w
  }
\cs_new_protected:Npn \__enumext_starred_item_viii_aux_ii:w
  {
    \legacy_if_set_true:n { @noitemarg }
    \__enumext_start_item_viii:w [ \l__enumext_label_viii_tl ]
  }
```

(*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.

```
\cs_new_protected:Nn \__enumext_starred_item_exec:
  {
    \tl_put_left:Ne \l__enumext_store_current_label_tl { \l__enumext_label_viii_tl }
    \__enumext_store_addto_prop:V \l__enumext_store_current_label_tl
    \__enumext_keyans_store_ref:
    \tl_put_left:Ne \l__enumext_store_current_label_tl { \item }
    \__enumext_keyans_addto_seq_link:
    \int_gincr:N \g__enumext_check_starred_cmd_int
    \bool_if:NT \l__enumext_show_answer_bool
      {
        \__enumext_print_keyans_box:NN \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
      }
    \bool_if:NT \l__enumext_show_position_bool
      {
        \tl_set:Ne \l__enumext_mark_answer_sym_tl
          {
            \group_begin:
              \exp_not:N \normalfont
              \exp_not:N \footnotesize [ \int_eval:n
                {
                  \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
                }
              ]
            \group_end:
          }
        \__enumext_print_keyans_box:NN \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
      }
  }
```

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

11.43.2 Real definition of `\item` in keyans*

`\__enumext_start_item_viii:w` The implementation at this point is very similar to that of the enumext* environment.

```
\cs_new_protected_nopar:Npn \__enumext_start_item_viii:w [#1]
  {
    \cs_set_eq:NN \__enumext_stop_item_tmp_viii: \__enumext_stop_item_viii:
    \legacy_if:nT { @noitemarg }
      {
        \legacy_if_set_false:n { @noitemarg }
        \legacy_if:nT { @nmbrlist }
          {
            \bool_if:NT \l__enumext_hyperref_bool
              {
                \legacy_if_set_true:n { @hyper@item }
              }
            \refstepcounter{enumXviii}
          }
      }
```

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

```
    \group_begin:
      \lrbox{ \l__enumext_item_text_viii_box }
        \bool_if:NF \l__enumext_footnotes_key_bool
          {
            \__enumext_renew_footnote:
          }
        \bool_if:NT \l__enumext_item_starred_viii_bool
          {
            \__enumext_starred_item_exec:
          }
        \group_begin:
          \tl_use:N \l__enumext_label_font_style_viii_tl
          \bool_if:NTF \l__enumext_wrap_label_viii_bool
            {
              \makebox[ \l__enumext_labelwidth_viii_dim ][ \l__enumext_align_label_viii_str ]
                { \__enumext_wrapper_label_viii:n {#1} }
            }
            {
              \makebox[ \l__enumext_labelwidth_viii_dim ][ \l__enumext_align_label_viii_str ]{ #1
            }
        \group_end:
        \skip_horizontal:N \l__enumext_labelsep_viii_dim
        \tl_use:N \l__enumext_after_list_args_viii_tl
        \__enumext_minipage:w [ t ]{ \l__enumext_joined_width_viii_dim  }
          \skip_set_eq:NN \parindent \l__enumext_listparindent_viii_dim
          \skip_set_eq:NN \parskip \l__enumext_parsep_viii_skip
          \bool_if:NT \l__enumext_item_starred_viii_bool
            {
              \tl_use:N \l__enumext_fake_item_indent_viii_tl
              \__enumext_keyans_show_item_opt:
              \skip_horizontal:n { -\l__enumext_fake_item_indent_viii_dim - \l__enumext_labelsep_
            }
            {
              \tl_use:N \l__enumext_fake_item_indent_viii_tl
            }
   }
```

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

`\__enumext_stop_item_viii:` The function `\__enumext_stop_item_viii:` shall terminate with the capture of `\item` and its ⟨*contents*⟩. Close the environments `minipage`, `lrbox` and the group. Then we only have to set the width of the box and print it next to `\footnote`, and add the horizontal and vertical separation between the boxes.

```
\cs_new_protected_nopar:Nn \__enumext_stop_item_viii:
  {
      \__enumext_endminipage:
    \endlrbox
  \group_end:
  \box_set_wd:Nn \l__enumext_item_text_viii_box
    {
      \l__enumext_joined_width_viii_dim
      + \l__enumext_labelwidth_viii_dim
      + \l__enumext_labelsep_viii_dim
    }
  \int_set:Nn \hbadness { 10000 }
  \box_use_drop:N \l__enumext_item_text_viii_box
  \bool_if:NF \l__enumext_footnotes_key_bool
    {
      \__enumext_print_footnote:
    }
  \int_compare:nNnTF
    { \l__enumext_item_column_pos_viii_int } = { \l__enumext_columns_viii_int }
    {
      \par\noindent
      \int_zero:N \l__enumext_item_column_pos_viii_int
    }
    { \hspace{ \l__enumext_columns_sep_viii_dim } }
  }
```

(*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.

```
\cs_new_protected:Nn \__enumext_remove_extra_parsep_viii:
  {
    \int_compare:nNnT
      {
        \int_mod:nn
          { \g__enumext_item_count_all_viii_int }
          { \l__enumext_columns_viii_int }
      }
      =
      { 0 }
      {
        \par
        \vspace{ -\l__enumext_itemsep_viii_skip }
        \int_gzero:N \g__enumext_item_count_all_viii_int
      }
  }
```

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

11.44 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.

```
\NewDocumentCommand \getkeyans { m }
  {
    \exp_args:Ne \__enumext_getkeyans_aux:n
      { \tl_to_str:e { \text_expand:n {#1} } }
  }
```

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

`\__enumext_getkeyans_aux:n` The internal function `\__enumext_getkeyans_aux:n` is in charge of *splitting* the ⟨*argument*⟩ using ":". If ":" is omitted it will return an error.

```
\cs_new_protected:Npn \__enumext_getkeyans_aux:n #1
  {
    \str_if_in:nnTF {#1} { : }
      {
        \use:e
          {
            \cs_set:Npn \exp_not:N \__enumext_tmp:w ##1 \c_colon_str ##2 \scan_stop:
              { {##1} {##2} }
          }
        \exp_after:wN \__enumext_getkeyans:nn \__enumext_tmp:w #1 \scan_stop:
      }
      { \msg_error:nnn { enumext } { missing-colon } {#1} }
  }
```

(*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*⟩.

```
\cs_new_protected:Npn \__enumext_getkeyans:nn #1 #2
  {
    \prop_if_exist:cF { g__enumext_#1_prop }
      { \msg_error:nnn { enumext } { undefined-storage-anskey } {#1} }
    \group_begin:
      \prop_item:cn { g__enumext_#1_prop }{#2}
    \group_end:
  }
```

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

11.45 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*⟩`]`

```
\cs_generate_variant:Nn \keys_precompile:nnN { neN }
\keys_define:nn  { enumext / print }
  {
    print*  .code:n     = \keys_precompile:neN { enumext / enumext* }
                            { \__enumext_filter_save_key:n {#1} }
                            \l__enumext_print_keyans_starred_tl, % starred cmd
    print*  .initial:n  = { nosep, label=\arabic*., columns=2, first=\small, font=\small },
    print-1 .code:n     = \keys_precompile:neN { enumext / level-1 }
                            { \__enumext_filter_save_key:n {#1} }
                            \l__enumext_print_keyans_i_tl,
    print-1 .initial:n  = { nosep, label=\arabic*., columns=2, first=\small, font=\small },
    print-2 .code:n     = \keys_precompile:neN { enumext / level-2 }
                            { \__enumext_filter_save_key:n {#1} }
                            \l__enumext_print_keyans_ii_tl,
    print-2 .initial:n  = { nosep, label=(\alph*), first=\small, font=\small },
    print-3 .code:n     = \keys_precompile:neN { enumext / level-3 }
                            { \__enumext_filter_save_key:n {#1} }
                            \l__enumext_print_keyans_iii_tl,
    print-3 .initial:n  = { nosep, label=\roman*., first=\small, font=\small },
    print-4 .code:n     = \keys_precompile:neN { enumext / level-4 }
                            { \__enumext_filter_save_key:n {#1} }
                            \l__enumext_print_keyans_iv_tl,
    print-4 .initial:n  = { nosep, label=\Alph*., first=\small, font=\small },
    print-* .code:n     = \keys_precompile:neN { enumext / enumext* }
                            { \__enumext_filter_save_key:n {#1} }
                            \l__enumext_print_keyans_vii_tl, % starred nested
    print-* .initial:n  = { nosep, label=\arabic*., first=\small, font=\small },
  }
```

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`.

```
\NewDocumentCommand \printkeyans { s O{} m }
  {
    \group_begin:
      \tl_use:N \l__enumext_print_keyans_i_tl
      \tl_use:N \l__enumext_print_keyans_ii_tl
      \tl_use:N \l__enumext_print_keyans_iii_tl
      \tl_use:N \l__enumext_print_keyans_iv_tl
      \tl_use:N \l__enumext_print_keyans_vii_tl
      \__enumext_printkeyans:nnn { #1 } { #2 } { #3 }
    \group_end:
  }
```

(*End of definition for* `\printkeyans`*. This function is documented on page 16.*)

`\__enumext_printkeyans:nnn`

The internal function `\__enumext_printkeyans:nnn` will check for the existence of the ⟨*sequence*⟩, if it does not exist it will return an error message, then it will check if not empty.

```
\cs_new_protected:Npn \__enumext_printkeyans:nnn #1 #2 #3
  {
    \seq_if_exist:cTF { g__enumext_#3_seq }
      {
        \seq_if_empty:cF { g__enumext_#3_seq }
          {
            %%\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*⟩.

```
            \bool_if:nTF {#1}
              {
                \seq_if_in:cnTF { g__enumext_#3_seq } { \end{enumext*} }
                  {
                    \msg_error:nnnn { enumext } { print-starred } {#3} { enumext* }
                  }
                  {
                    \tl_use:N \l__enumext_print_keyans_starred_tl
                    \begin{enumext*}[#2]
                      \keys_set:nn { enumext / level-1 }{ base-fix }
                      \seq_map_inline:cn { g__enumext_#3_seq } { ##1 }
                    \end{enumext*}
                  }
              }
```

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*⟩.

```
              {
                \begin{enumext}[#2]
                  \keys_set:nn { enumext / enumext* }{ base-fix }
                  \seq_map_inline:cn { g__enumext_#3_seq } { ##1 }
                \end{enumext}
              }
          }
      }
      {
        \msg_error:nnn { enumext } { undefined-storage-anskey } {#3}
      }
  }
```

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

11.46 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.

```
\cs_new:Npn \__enumext_filter_first_level:n #1
  {
    \use:e
      {
        \keyval_parse:NNn
          \__enumext_filter_first_level_key:n
          \__enumext_filter_first_level_pair:nn {#1}
      }
  }
```

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*`.

```
\cs_new:Npn \__enumext_filter_first_level_key:n #1
  {
    \str_case:nnF {#1}
      {
        { resume   } {}
        { resume*  } {}
      }
      { , { \exp_not:n {#1} } }
  }
```

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.

```
\cs_new:Npn \__enumext_filter_first_level_pair:nn #1#2
  {
    \str_case:nnF {#1}
      {
        { series } {}
        { resume } {}
        { save-ans } {}
      }
      { , { \exp_not:n {#1} } = { \exp_not:n {#2} } }
  }
```

(*End of definition for* `\__enumext_filter_first_level:n`, `\__enumext_filter_first_level_key:n`, *and* `\__enumext_filter_first_level_pair:nn`.)

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

```
\keys_define:nn { enumext / meta-families }
  {
    enumext-1 .code:n =
                  {
                    \keys_set:ne { enumext / level-1  }
                      {
                        \__enumext_filter_first_level:n {#1}
                      }
                  } ,
    enumext-2 .code:n = { \keys_set:nn { enumext / level-2  } {#1} } ,
    enumext-3 .code:n = { \keys_set:nn { enumext / level-3  } {#1} } ,
    enumext-4 .code:n = { \keys_set:nn { enumext / level-4  } {#1} } ,
    keyans    .code:n = { \keys_set:nn { enumext / keyans   } {#1} } ,
    enumext*  .code:n =
                  {
                    \keys_set:ne { enumext / enumext* }
                      {
                        \__enumext_filter_first_level:n {#1}
                      }
                  } ,
    keyans*   .code:n = { \keys_set:nn { enumext / keyans*  } {#1} } ,
    print*    .code:n = { \keys_set:nn { enumext / print    } { print*  = {#1} } } ,
    print-1   .code:n = { \keys_set:nn { enumext / print    } { print-1 = {#1} } } ,
    print-2   .code:n = { \keys_set:nn { enumext / print    } { print-2 = {#1} } } ,
    print-3   .code:n = { \keys_set:nn { enumext / print    } { print-3 = {#1} } } ,
    print-4   .code:n = { \keys_set:nn { enumext / print    } { print-4 = {#1} } } ,
    print-*   .code:n = { \keys_set:nn { enumext / print    } { print-* = {#1} } } ,
    unknown   .code:n = { \msg_error:nn { enumext } { unknown-key-family } } ,
  }
```

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

```
\seq_const_from_clist:Nn \c__enumext_all_families_seq
  {
    enumext-1, enumext-2, enumext-3, enumext-4, keyans, enumext*,
    keyans*, print-1, print-2, print-3, print-4, print-*, print*,
  }
```

`\setenumext` Now we define the user command `\setenumext`.

```
\NewDocumentCommand \setenumext { O{enumext,1} +m }
  {
    \tl_if_novalue:nTF {#1}
```

```
      {
        \seq_map_inline:Nn \c__enumext_all_families_seq
      }
      {
        \seq_clear:N \l__enumext_setkey_tmpa_seq
        \seq_set_from_clist:Nn \l__enumext_setkey_tmpb_seq {#1}
        \int_set:Nn \l__enumext_setkey_tmpa_int
          {
            \seq_count:N \l__enumext_setkey_tmpb_seq
          }
        \int_compare:nNnTF { \l__enumext_setkey_tmpa_int } > { 1 }
          {
            \seq_pop_left:NN \l__enumext_setkey_tmpb_seq \l__enumext_setkey_tmpa_tl
            \seq_map_function:NN \l__enumext_setkey_tmpb_seq \__enumext_set_parse:n
            \seq_set_map_e:NNn \l__enumext_setkey_tmpa_seq \l__enumext_setkey_tmpa_seq
              {
                \tl_use:N \l__enumext_setkey_tmpa_tl - ##1
              }
          }
          {
            \seq_put_right:Ne \l__enumext_setkey_tmpa_seq { \tl_trim_spaces:n {#1} }
          }
        \seq_if_empty:NTF \l__enumext_setkey_tmpa_seq
          { \seq_map_inline:Nn \c__enumext_all_families_seq }
          { \seq_map_inline:Nn \l__enumext_setkey_tmpa_seq }
      }
      {
        \keys_set:nn { enumext / meta-families } { ##1 = {#2} }
      }
  }
```

(*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.

```
\cs_new_protected:Npn \__enumext_set_parse:n #1
  {
    \tl_set:Ne \l__enumext_setkey_tmpb_tl { \tl_trim_spaces:n {#1} }
    \clist_map_inline:nn { 0, 1, 2, 3, 4, * } % <- max level
      { \tl_remove_all:Nn \l__enumext_setkey_tmpb_tl {##1} }
    \tl_if_empty:NTF \l__enumext_setkey_tmpb_tl
      {
        \seq_put_right:Ne \l__enumext_setkey_tmpa_seq
          { \tl_trim_spaces:n {#1} }
      }
      { \__enumext_set_error:nn {#1} { } }
  }
\cs_new_protected:Npn \__enumext_set_error:nn #1 #2
  { \msg_error:nnn { enumext } { invalid-key } {#1} {#2} }
```

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

11.47 Messages

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

```
\msg_new:nnn { enumext } { package-load }
  {
    The ~ '#1' ~ package ~ is ~ already ~ loaded.
  }
\msg_new:nnn { enumext } { package-not-load }
  {
    The ~ '#1' ~ package ~ will ~ be ~ loaded ~ as ~ a ~ dependency.
  }
\msg_new:nnn { enumext } { package-load-foot }
  {
    The ~ '#1' ~ package ~ is ~ loaded ~ with ~ the ~ option ~ '#2'.
  }
```

Message used in the creation of counters by enumext package.

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

Message used by align and mark-pos keys.

```
\msg_new:nnn { enumext } { unknown-choice }
  {
    The ~ value ~ '#3' ~ for ~ '#1' ~ key ~ is ~ invalid ~ use ~ ('#2').
  }
```

Message used by reserved anskey* environment by enumext package.

```
\msg_new:nnnn { enumext } { anskey-env-error }
  {
    The ~ '#1' ~ environment ~is ~ reserved ~ by ~\\
    'enumext' ~ package, ~ It~ is~ already~ defined.
  }
  {
    The ~ anskey* ~ environment ~ is ~ defined ~ internally ~
    for ~ the ~ 'save-ans' ~ key.\\
  }
```

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

```
\msg_new:nnn { enumext } { store-prop }
  {
    * ~ Package ~ enumext: ~ Creating ~
     \c_backslash_str g__enumext_#1_prop ~ \msg_line_context:.
  }
\msg_new:nnn { enumext } { store-seq }
  {
    * ~ Package ~ enumext: ~ Creating ~
    \c_backslash_str g__enumext_#1_seq ~ \msg_line_context:.
  }
\msg_new:nnn { enumext } { store-int }
  {
    * ~ Package ~ enumext: ~ Creating ~
    \c_backslash_str g__enumext_resume_#1_int ~ \msg_line_context:.
  }
\msg_new:nnn { enumext } { prop-seq-int-hook }
  {
    * ~ Package ~ enumext: ~ Elements ~ in ~
    \c_backslash_str g__enumext_#1_prop ~ = ~ #2.\\
    * ~ Package ~ enumext: ~ Elements ~ in ~
    \c_backslash_str g__enumext_#1_seq ~ = ~ #3.\\
    * ~ Package ~ enumext: ~ Value ~ off ~
    \c_backslash_str g__enumext_resume_#1_int ~ = ~ #4.
  }
\msg_new:nnn { enumext } { item-answer-hook }
  {
    * ~ Package ~ enumext: ~ Value ~ off ~
    \c_backslash_str g__enumext_item_number_int ~ = ~ #1.\\
    * ~ Package ~ enumext: ~ Value ~ off ~
    \c_backslash_str g__enumext_item_anskey_int ~ = ~ #2.\\
    * ~ Package ~ enumext: ~ Difference ~ item_number_int ~ - ~ item_anskey_int ~ = ~ #3.
  }
```

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

```
\msg_new:nnn { enumext } { invalid-key }
  {
    The ~ key ~ '#1' ~ is ~ not ~ know ~ the ~ level ~ #2.
  }
\msg_new:nnn { enumext } { unknown-key-family }
  {
    Unknown~key~family~`\l_keys_key_str'~for~enumext.
  }
```

Messages used in length calculation.

```
\msg_new:nnn { enumext } { width-negative }
  {
    Ignoring ~ negative ~ value ~ '#1=#2' ~ \msg_line_context:.\\
    The ~ key ~ '#1'~ accepts ~ values  ~ >= ~ 0pt.
  }
\msg_new:nnn { enumext } { width-zero }
  {
    Invalid ~ '#1=#2' ~ \msg_line_context:.\\
    The ~ key ~ '#1'~ accepts ~ values  ~ > ~ 0pt.
  }
```

Messages used by show-length key in enumext.

```
\msg_new:nnn { enumext } { list-lengths }
  {
    **** ~ Lengths ~ used ~ by ~ 'enumext' ~ level ~ '#2' ~ \msg_line_context:~\c_space_tl ****\\
    \__enumext_show_length:nnn { dim  } { labelsep      } {#1}
    \__enumext_show_length:nnn { dim  } { labelwidth    } {#1}
    \__enumext_show_length:nnn { dim  } { itemindent    } {#1}
    \__enumext_show_length:nnn { dim  } { leftmargin    } {#1}
    \__enumext_show_length:nnn { dim  } { rightmargin   } {#1}
    \__enumext_show_length:nnn { dim  } { listparindent } {#1}
    \__enumext_show_length:nnn { skip } { topsep    } {#1}
    \__enumext_show_length:nnn { skip } { parsep    } {#1}
    \__enumext_show_length:nnn { skip } { partopsep } {#1}
    \__enumext_show_length:nnn { skip } { itemsep   } {#1}
    ****************************************************
  }
```

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

```
\msg_new:nnn { enumext } { list-lengths-not-nested }
  {
    **** ~ Lengths ~ used ~ by ~ '#2' ~ environment ~ \msg_line_context:~\c_space_tl ****\\
    \__enumext_show_length:nnn { dim  } { labelsep      } {#1}
    \__enumext_show_length:nnn { dim  } { labelwidth    } {#1}
    \__enumext_show_length:nnn { dim  } { itemindent    } {#1}
    \__enumext_show_length:nnn { dim  } { leftmargin    } {#1}
    \__enumext_show_length:nnn { dim  } { rightmargin   } {#1}
    \__enumext_show_length:nnn { dim  } { listparindent } {#1}
    \__enumext_show_length:nnn { skip } { topsep    } {#1}
    \__enumext_show_length:nnn { skip } { parsep    } {#1}
    \__enumext_show_length:nnn { skip } { partopsep } {#1}
    \__enumext_show_length:nnn { skip } { itemsep   } {#1}
    ****************************************************
  }
```

Messages used by ref key.

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

Messages used by save-ans key.

```
\msg_new:nnn { enumext } { save-ans-empty }
  {
    Key ~ 'save-ans' ~ need ~ a ~ value ~ in ~ '#1'~ \msg_line_context:.
  }
\msg_new:nnn { enumext } { save-ans-log }
  {
    * ~ Package ~ enumext: ~ Start ~ #1\c_space_tl with ~ save-ans=#2 ~ \msg_line_context:.
  }
\msg_new:nnn { enumext } { save-ans-log-hook }
  {
    * ~ Package ~ enumext: ~ Stop ~ #1\c_space_tl with ~ save-ans=#2 ~ \msg_line_context:.
  }
\msg_new:nnn { enumext } { save-ans-hook }
  {
    Stop ~ storing ~ for ~ 'save-ans=#1' ~ \msg_line_context:.
  }
```

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

```
\msg_new:nnn { enumext } { need-save-ans }
  {
    Key ~ '#1'~ works ~ only ~ with ~ the ~ 'save-ans' ~ key ~ in ~ '#2'~ \msg_line_context:.
  }
\msg_new:nnn { enumext } { items-same-answer }
  {
    ****************************************\\
    * ~ Package ~ enumext: ~ Checking ~ answers ~ in ~ '#1' ~
    for ~ \c_left_brace_str #2 \c_right_brace_str\\
    * ~ started ~ #3 ~ and ~ close ~ \msg_line_context: : ~
    'OK', ~ all ~ items ~ with ~ answer.\\
    ****************************************
  }
\msg_new:nnn { enumext } { item-greater-answer }
```

```
  {
    Checking ~ answers ~ in ~ '#1' ~ for ~ \c_left_brace_str #2 \c_right_brace_str\\
    started ~ #3 ~ and ~ close ~ \msg_line_context: : ~'NOT ~ OK'\\
    Items ~ > ~ Answers.
  }
\msg_new:nnn { enumext } { item-less-answer }
  {
    Checking ~ answers ~ in ~ '#1' ~ for ~ \c_left_brace_str #2 \c_right_brace_str\\
    started ~ #3 ~ and ~ close ~ \msg_line_context: : ~'NOT ~ OK'\\
    Items ~ < ~ Answers.
  }
```

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

```
\msg_new:nnn { enumext } { missing-starred }
  {
    Missing ~ '\c_backslash_str #1*' ~ #2.
  }
\msg_new:nnn { enumext } { many-starred }
  {
    Many ~ '\c_backslash_str #1*' ~ #2.
  }
```

Messages used by `\printkeyans*` command.

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

Message for the nesting depth of the environment `enumext`.

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

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

```
\msg_new:nnn { enumext } { anskey-unnumber-item }
  {
    Can't ~ store ~ with ~ a ~ unnumbered ~ \c_backslash_str item ~ \msg_line_context:.
  }
\msg_new:nnn { enumext } { anskey-already-stored }
  {
    Content ~ already ~ stored ~ for ~ this ~ \c_backslash_str item ~ \msg_line_context:.
  }
\msg_new:nnn { enumext } { anskey-empty-arg }
  {
    Can't ~ store ~ empty ~ content ~ ~ \msg_line_context:.
  }
\msg_new:nnn { enumext } { anskey-wrong-place }
  {
    Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:.~ \\
    '\c_backslash_str #1' ~ works ~ in ~ the ~ environment ~ '#2'.
  }
\msg_new:nnn { enumext } { anskey-nested }
  {
    The ~ command ~ \c_backslash_str anskey~ can't ~ be ~ nested ~ \msg_line_context:.
  }
\msg_new:nnn { enumext } { anskey-math-mode }
  {
    #1 ~ can't ~ work ~ in ~ math ~ mode ~ \msg_line_context:.
  }
\msg_new:nnn { enumext } { anskey-env-wrong }
  {
    The ~ environment ~ anskey* ~ cannot ~ use ~ in ~ '#1' ~ \msg_line_context:.
  }
\msg_new:nnn { enumext } { anspic-wrong-place }
  {
    Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:.~ \\
    '\c_backslash_str #1' ~ works ~ in ~ the ~ environment ~ '#2'.
  }
\msg_new:nnn { enumext } { command-wrong-place }
  {
    Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:.~ \\
    '\c_backslash_str #1' ~ works ~ outside ~ the ~ environment ~ '#2'.
  }
\msg_new:nnnn { enumext } { anskey-env-key-unknown }
  {
    The ~ key ~ '#1' ~ is ~ unknown ~ by ~ environment~
    'anskey*' ~ and ~ is ~ being ~ ignored.
  }
  {
    The ~ environment ~ 'anskey*' ~ does ~ not ~ have ~ a ~ key ~ called ~'#1'.\\
    Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
  }
\msg_new:nnnn { enumext } { anskey-env-key-value-unknown }
  {
    The ~ key ~ '#1=#2' ~ is ~ unknown ~ by ~ environment ~
    'anskey*' ~ and ~ is ~ being ~ ignored.
  }
  {
    The ~ environment ~ 'anskey*' ~ does ~ not ~ have ~ a ~ key ~ called ~'#1'.\\
    Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
  }
\msg_new:nnnn { enumext } { anskey-cmd-key-unknown }
  { The~key~'#1'~is~unknown~by~'\c_backslash_str anskey'~and~is~being~ignored.}
  {
    The~command~'\c_backslash_str anskey'~does~not~have~a~key~called~'#1'.\\
    Check~that~you~have~spelled~the~key~name~correctly.
  }
\msg_new:nnnn { enumext } { anskey-cmd-key-value-unknown }
  { The ~ key ~ '#1=#2' ~ is ~ unknown ~ by ~ '\c_backslash_str anskey' ~ and ~ is ~ being ~ igno
  {
    The ~ command ~ '\c_backslash_str anskey' ~ does ~ not ~ have ~ a ~ key ~ called ~'#1'.\\
    Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
  }
```

Messages used by keyans and keyanspic environment.

```
\msg_new:nnn { enumext } { keyans-nested }
  {
    The ~ environment ~ 'keyans' ~ can't ~ be  ~ nested  ~ \msg_line_context:.
  }
\msg_new:nnn { enumext } { keyans-wrong-level }
  {
    Wrong ~ level ~ position ~ for ~ 'keyans' ~ \msg_line_context:.~ \\
    The ~ environment ~ 'keyans' ~ can ~ only ~ be ~ in ~ the ~ first ~ level.
  }
\msg_new:nnn { enumext } { wrong-place }
  {
    Wrong ~ place ~ for ~ '#1' ~ environment ~\msg_line_context:.~ \\
    '#1' ~ is ~ only ~ found ~ with ~ '#2' ~  in  ~  'enumext.
  }
\msg_new:nnn { enumext } { keyanspic-nested }
  {
    The ~ environment ~ 'keyanspic' ~ can't ~ be  ~ nested~ \msg_line_context:.~.
  }
\msg_new:nnn { enumext } { keyanspic-wrong-level }
  {
    Wrong ~ level ~ position ~ for ~ 'keyanspic' ~ \msg_line_context:.~ \\
    The ~ environment ~ 'keyans' ~ can ~ only ~ be ~ in ~ the ~ first ~ level.
  }
```

Messages used by \getkeyans command.

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

Messages used by \miniright command.

```
\msg_new:nnn { enumext } { missing-miniright }
  {
    Missing ~ '\c_backslash_str miniright' ~ in ~ \msg_line_context:.\\
    The ~ key ~ 'mini-env' ~ need ~ '\c_backslash_str miniright'.
  }
\msg_new:nnn { enumext } { wrong-miniright-place }
```

```
  {
    Wrong ~ place ~ for ~ '\c_backslash_str miniright' ~ \msg_line_context:.~ \\
    Works ~ in ~ 'enumext' ~ and ~ 'keyans' ~ with ~ key ~ 'mini-env'.
  }
\msg_new:nnn { enumext } { wrong-miniright-use }
  {
    Wrong ~ use ~ for ~ '\c_backslash_str miniright' ~ \msg_line_context:.~ \\
    '\c_backslash_str miniright' ~ need ~ a ~ key ~ 'mini-env'.
  }
```

Messages used by `enumext*` and `keyans*` environments.

```
\msg_new:nnn { enumext } { nested }
  {
    The ~ starred ~ environment ~ can't ~ be ~ nested ~ \msg_line_context:.
  }
\msg_new:nnn { enumext } { item-joined }
  {
    Items ~ joined ~ (#1) ~ > ~ #2  ~ columns ~\msg_line_context:.
  }
\msg_new:nnn { enumext } { item-joined-columns }
  {
    Not ~ space ~ to ~ join ~ items ~ (#1) ~ > ~ #2 ~\msg_line_context:.
  }
```

11.48 Finish package

Finish package implementation.

```
\file_input_stop:
⟨/package⟩
```

12 Index of Implementation

The italic numbers denote the pages where the corresponding entry is described, the numbers underlined and all others indicate the line on which they are implemented in the package code.