enumext

ENUMERATE EXERCISE SHEETS

v1.7 2025-07-10*

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

Abstract

This package provides enumerated list environments compatible with *tagging* PDF for creating *"simple exercise sheets"* along with *"multiple choice questions"*, storing the *"answers"* to these in memory using multicol package.

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 v1.7, last revised 2025-07-10.
†E-mail: «pablgonz@educarchile.cl».

License and Requirements

The enumext package loads and requires multicol[3] package, 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.

The minimum requirement is LATEX release 2025-06-01.

1 Introduction

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

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

1. Factor $x^2 - 2x + 1$
2. Factor $3x + 3y + 3z$
3. True False
   (a) $\alpha > \delta$
   (b) LATEX2e is cool?
4. Related to Linux

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

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

1. Factor $x^2 - 2x + 1$
   * $(x-1)^2$
2. Factor $3x + 3y + 3z$
   * $3(x+y+z)$
3. True False
   (a) $\alpha > \delta$
      * False
   (b) LATEX2e is cool?
      * Very True!
4. Related to Linux

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

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

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

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

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

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

1. First type of questions

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

2. Second type of questions

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

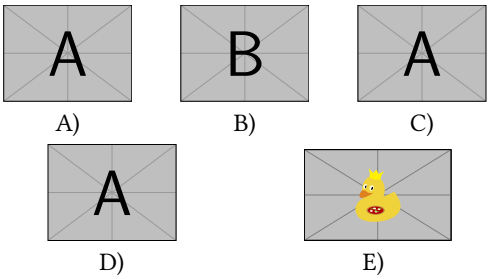
   A) I only
   B) II only
   C) I and II only
   D) I and III only
   E) I, II, and III

⋆ 3. Third type of questions

   (1) $2\alpha + 2\delta = 90°$
   (2) $\angle EDF = 45°$

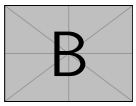
   A) value
   B) value
   C) value
   D) value
   E) value

4. Question with image and label below:



5. Question with image on right 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"

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

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.

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 `luatex enumext.ins` and move all files to appropriate locations, then run `mktexlsr`. To produce the documentation run `arara enumext.dtx`.

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

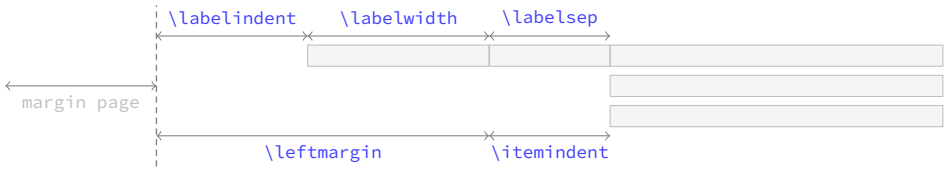


Figure 1: Representation of horizontal lengths in `enumitem`.

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.

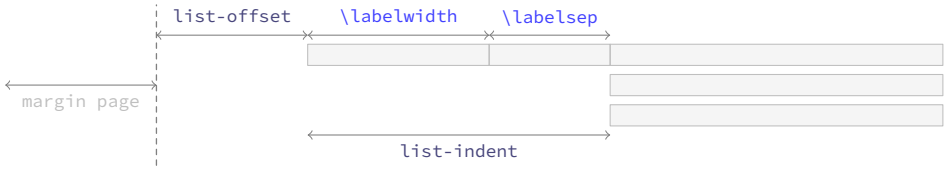


Figure 2: Representation of horizontal lengths concept in enumext.

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.

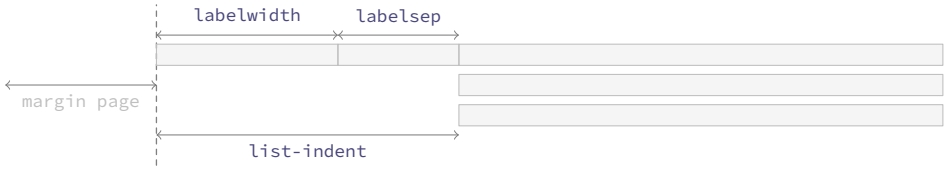


Figure 3: Default horizontal lengths `list-offset=0pt`, `list-indent=\labelwidth+\labelsep` in enumext.

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*, \foreachkeyans to print a rage of *stored content*, \miniright for minipage, \setenumext to config [⟨*key = val*⟩], \setenumextmeta to add a *"meta-key"* and \resetenumext for reset counters.

1.3.1 Public counters

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

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

1.3.2 Public dimension

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

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

1.3.3 Support for multicol

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

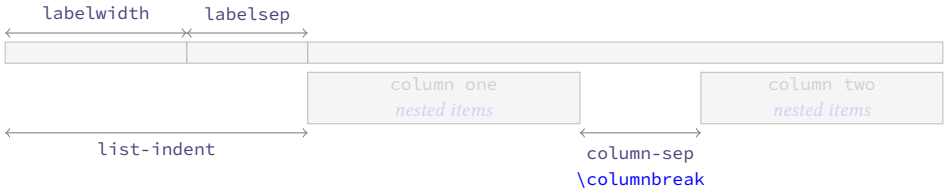


Figure 4: Representation of the two column output for a nested level in enumext environment.

The *"non starred"* version of the multicols environment is always used together with the \raggedcolumns command and is controlled by columns and columns-sep keys. It can be used in all nesting levels of the environment enumext and the environment keyans and can together with the mini-env key. If you need to force a start a new column \columnbreak must be used (see §5.7).

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

1.3.4 Support for minipage

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

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

The minipage environments on *"left side"* and *"right side"* is always used with *"aligned on top"* [t]. It can be used in all nesting levels of the environment enumext and the environment keyans and is controlled by mini-env and mini-sep keys. In order to switch from the *"left"* side minipage environment to the *"right"* side one must use the command \miniright (see §5.8).

1.3.5 The \label and \ref system

This package provides a user interface like the enumitem[6] package to customize the references which is activated by the ref key (§5.1), the standard LaTeX \label and \ref commands work as usual. It also provides an *"internal reference"* system for the *"stored content"* by means of the key save-ref (§6.1.1) when the key save-ans is active (§6.1).

1.3.6 Support for \footnote

The enumext* and keyans* environments and the mini-env key use the minipage environment in their implementation but in a transparent way for the user, it is only used for typesetting. The implementation supports \footnote and is compatible with hyperref and works the same way as if used anywhere in the document.

Unfortunately, if hyperref is loaded and *tagged* PDF is not active, it will not produce the "link's", the internal implementation uses \footnotetext[⟨*number*⟩] and \footnotemark[⟨*number*⟩]{⟨*text*⟩} and "link's" for this are not supported by the hyperref package.

The best way to solve this if *tagged* PDF is NOT active is to use Jean-François Burnol footnotehyper[9] package, it will support keeping the "link's" if hyperref is loaded with the hyperfootnotes=true option (default). Load it is as follows:

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

At the moment the footnotehyper package v1.1e (2021/08/13) is not compatible with *tagged* PDF.

2 The environments provided

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

enumext
enumext*

```
\begin{enumext}[⟨keyval list⟩]
  \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 shortenumerate or tasks environments provided by the shortlst[16] and tasks[17] packages, \item and \item[⟨*custom*⟩] work as usual. The options can be configured globally using \setenumext command and locally using [⟨*key* = *val*⟩] in the environment.

Some considerations to take into account for this environment:

- The environment cannot be nested within itself or in the environment keyans*, but it can be nested within enumext and vice versa.
- Each *"item content"* in the environment is placed within a minipage environment whose *width* is stored in the dimension \itemwidth that NOT includes labelwith, labelsep, only the *width of the content.*
- You cannot have floating environments like figure or table or \marginpar but \footnote with hyperref is supported. If you want to activate link's *"without" tagged* PDF active you must load footnotehyper and hyperref packages (see §1.3.6).
- You cannot have any standard list environments like itemize, enumerate, description, quote, quotation, verse, center, flushleft, flushright, verbatim, tabbing, trivlist, list and all environments created with \newtheorem.

Example with columns=2

1. This text is in the first level.
X This text is in the first level.
2. This text is in the first level.
⋆ 4. This text is in the first level.

2.3 The command \item*

\item* \item*[⟨*symbol*⟩][⟨*offset*⟩]

The \item*, \item*[⟨*symbol*⟩] and \item*[⟨*symbol*⟩][⟨*offset*⟩] works like the numbered \item, but placing a ⟨*symbol*⟩ to the *"left"* of the ⟨*label*⟩ separated from it by the ⟨*offset*⟩ set by the the *second optional argument.*

The *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: \textborn

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*={$\star$}.

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

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

1. The first
⋆ 2. The second
3. The third
4. The fourth
⋆ 5. The fifth item is way too long for this and needs three columns
6. The sixth
7. The seventh
X The eighth item is way too long for this and needs two columns (196.17749pt)
9. The ninth
Z The tenth (89.28171pt)

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 argument* 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.

For security reasons the keys resume *"with value"*, resume*, reset, reset*, series and save-ans they can NOT be set by this command and are ignored. The key save-ans that activate the *"storage system"* must be passed directly in the *optional argument* of the *"first level"* of the environment in which they are executed.

4 The command \setenumextmeta

\setenumextmeta

\setenumextmeta[1]{⟨*key name*⟩}{⟨*key-one = val, key-two = val, ...*⟩}
\setenumextmeta[2]{⟨*key name*⟩}{⟨*key-one = val, key-two = val, ...*⟩}
\setenumextmeta[3]{⟨*key name*⟩}{⟨*key-one = val, key-two = val, ...*⟩}
\setenumextmeta[4]{⟨*key name*⟩}{⟨*key-one = val, key-two = val, ...*⟩}
\setenumextmeta[*]{⟨*key name*⟩}{⟨*key-one = val, key-two = val, ...*⟩}
\setenumextmeta*{⟨*key name*⟩}{⟨*key-one = val, key-two = val, ...*⟩}

The command \setenumextmeta adds a new *"meta-key"* for the environments enumext and enumext*, the {⟨*key name*⟩} must be different from those defined by the package. The *optional argument* of the form [1], [2], [3], [4] adds a new *"meta-key"* for levels 1, 2, 3 and 4 of the enumext environment, the form [*] adds a new *"meta-key"*for the enumext* environment. If is run *without* the *optional argument*, it will adds a new *"meta-key"* for the *"first level"* of the enumext environment.

The *starred argument* '*' will create the new *"meta-key"* for the environment enumext* and for *"all levels"* of the environment enumext. For example: \setenumextmeta*{midsep}{topsep=3pt, partopsep=0pt} will create a new key midsep available for all levels of the enumext environment and the enumext* environment and we can use it like any other key so \begin{enumext}[midsep] and \begin{enumext*}[midsep] will be valid.

The old syntax \setenumextmeta[⟨*enumext**⟩] and \setenumextmeta[⟨*enumext, level*⟩] is still supported but is not recommended and may be removed in the future.

5 The keyval system

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

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

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

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

5.1 Keys for label and ref

mode-box ⟨*value forbidden*⟩ default: *not used*

This is a *"switch-key"* that does not receive an argument and is "only" available for the *"first level"* of the enumext environment and the enumext* environment. When this is set the label, font, wrap-label and wrap-label* keys are executed within \makebox for the enumext and keyans environments.

This key is intended for compatibility with *tagged* PDF and is forcibly *"enabled"* when \DocumentMetadata is present. If you want to get the same document output whether \DocumentMetadata is active or not, you must enable this key.

In the enumext* and keyans* environments \makelabel are redefined using \makebox by default. If enumext or keyans is used in the enumext* environment the key must be activated manually.

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

Sets the ⟨*label*⟩ that will be printed at the *current level* and default value for labelwidth key. 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 label and ref"* 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, wrap-label and/or wrap-label* keys.

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

Sets the *horizontal space* between the box containing the current ⟨*label*⟩ defined by label key and the text of an item on the first line. Internally sets the value of \labelsep for the current level.

labelwidth = {⟨*rigid length*⟩} default: *by label*

Sets the *width* of the label box containing the current ⟨*label*⟩ set by the 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 set by label key 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 of width set by labelwidth.

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

Wraps the *current* ⟨*label*⟩ defined by label key referenced by {#1} after executing the align and font keys. The {⟨*code*⟩} must be passed between braces and this 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* '{##1}'. For example wrap-label={\fbox{#1}} or you can create a command:

```
\NewDocumentCommand \mywrap { s m }
  {
    \IfBooleanTF{#1}
      {\textcolor{red}{\textbf{Q}}\textcolor{blue}{\textbf{.}}\textcolor{gray}{#2}}
      {\textcolor{blue}{\textbf{Q}}\textcolor{red}{\textbf{.}}\textcolor{gray}{#2}}
  }
```

and then pass it through the key `wrap-label={\mywrap{#1}}` or `wrap-label={\mywrap*{#1}}`.

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

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

`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`.

5.2 Keys for penalties

Page breaks in the provided environments are controlled by the following three parameters, which work together to ensure they look good, avoiding unsightly page breaks that could distort the output.

`beginpenalty` = {⟨*integer*⟩} default: *-51*

Set the *page breaking* penalty for breaking at the beginning of the `enumext`, `enumext*`, `keyans`, and `keyans*` environments. Internally sets the value of `\@beginparpenalty`.

`midpenalty` = {⟨*integer*⟩} default: *-51*

Set the *page breaking* penalty for breaking between items of the `enumext`, `enumext*`, `keyans`, and `keyans*` environments. Internally sets the value of `\@itempenalty`.

`endpenalty` = {⟨*integer*⟩} default: *-51*

Set the *page breaking* penalty for breaking at the end of the `enumext`, `enumext*`, `keyans`, and `keyans*` environments. Internally sets the value of `\@endparpenalty`.

The values passed to these ⟨*keys*⟩ affect the nested environments in which they were set and cannot be reset. LaTeX default is `-\@lowpenalty`, that is, `-51`. Because it is negative, it somewhat encourages a page break at each spot. Change it with, e.g., `\@beginparpenalty=9999`; a value of `10000` prohibits a page break. Please, refer to your LaTeX or TeX manual about how penalties control page breaks.

5.3 Keys for spaces

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

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

5.3.1 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`.

In the `enumext*` and `keyans*` environments this value is passed to `\parskip` within the `minipage` environment where *"item content"* is placed.

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

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

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

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

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

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

In the enumext* and keyans* environments this value corresponds to the separation between rows.

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 *"switch-key"* that does not receive an argument available *only* for the *"first level"* of environment enumext. Fix the *baseline* when an environment enumext is nested in enumext* 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.

This key is provided as a way to work around this minor issue, but you should be aware that if for some reason you have the itemindent key set in the enumext* environment it will be lost and you will need to adjust it using the list-offset key in the enumext environment.

Extra vertical spaces

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 *"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 *"above"* the environment without hindering the value of the topsep key. The space is added with \vspace* so is *"not discardable"*.

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

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

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

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

5.3.2 Horizontal spaces

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 environments enumext and keyans 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"*.

The enumext* and keyans* environments are implemented using \makebox and minipage which causes *"list indent"* to always be equal to the value passed to labewdith plus labelsep. Passing a value to this key is equivalent to setting the value for the list-offset key.

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

Sets the extra *horizontal indentation*, beyond labelsep, of the *"first line"* off each \item that is not followed by a *"blank line"* or the \par command . This value must be greater than or equal to 0pt and is applied internally using \hspace *without* modifying the value of \itemindent.

This key is intended for the enumext* and keyans* environments where, by their implementation, it is not possible to adjust labelwidth and list-indent *without* modifying the output. If you use enumext or keyans and want to get around the *blank line* limitation or the \par command followed by \item you can modify labelwidth and list-indent and get the same effect.

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.

In the enumext* and keyans* environments this value is passed to \parindent within the minipage environment where *"item content"* is placed.

5.4 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"* all calculations related to the *list parameters* in the environment and the ⟨keys⟩ sets by [⟨key = val⟩] have been performed, with the exception of the ⟨keys⟩ start and start*, that is, in the *second argument* of the list: \begin{list}{⟨arg one⟩}{⟨arg two⟩{⟨code⟩}}.

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

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

first = {⟨code⟩} default: *not used*

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

Keep in mind that the {⟨code⟩} set in this ⟨key⟩ will affect the entire *"body"* of the environment and therefore the inner levels of the list and the keyans, keyans* and keyanspic environments. It is recommended to set this ⟨key⟩ per level. In the enumext* and keyans* environments this ⟨key⟩ is executed *"after"* the listparindent, parsep and itemindent ⟨keys⟩ within the minipage environment in which the *"item content"* is placed.

after = {⟨code⟩} default: *not used*

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

5.5 Keys for start, series and resume

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

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

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

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

For compatibility with *tagged* PDF, the *start value* are set *"after"* the *second argument* to the list environment and *"before"* the execution of the first \item and the first key: \begin{list}{⟨arg one⟩}{⟨arg two⟩}\setcounter{enumX}\item.

The following ⟨keys⟩ are available only for the enumext and enumext* environments.

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

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

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

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

Sets the *start value* and *options* for the *"current level"* continuing the numbering and *options* of the *"same level"* as the environment in which the `series={`⟨*series name*⟩`}` key was executed, the *start value* will continue numbering according to the last execution of `resume={`⟨*series name*⟩`}`. If passed *"without value"* this will only set *start value* continue the numbering of the *"same level"* from the last environment and level in which `series={`⟨*series name*⟩`}` or `resume={`⟨*series name*⟩`}` is NOT present and if the `save-ans` key is active (on the left) it will continue the numbering from the "last" environment in which it was executed. The *start value* can be overwritten using `start` or `start*` keys.

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*.

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

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

When using the key `resume={`⟨*series name*⟩`}` or `resume*` you will have hierarchy in the ⟨*keys*⟩ that are stored in {⟨*series name*⟩} or in an internal version of {⟨*series name*⟩} in the case of `resume*`. If you want to *reset* the value of a ⟨*key*⟩ that is already stored in {⟨*series name*⟩} or in an internal version of {⟨*series name*⟩} this must be placed to the *right* of the key `resume={`⟨*series name*⟩`}` or `resume*`.

When the `resume*` key is executed consecutively, it does not rewrite the ⟨*keys*⟩ stored in the internal version of {⟨*series name*⟩} and if the environment that precedes it does not have the *optional argument*, it will just continue with the numbering.

5.6 Keys for `reset`

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

Resets the *start value* of the *"counters"* in the `enumext` and `enumext*` environments along with the *"internal counters"* used by the `resume` *without value* and `resume*` keys at the *"level"* at which it is executed. The *start value* can be overwritten using the `start` or `start*` keys.

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

Resets the *start value* of the *"counters"* in the `enumext` and `enumext*` environments along with the *"internal counters"* used by the `resume` *without value* and `resume*` keys at the *"level"* at which it is executed and in the *"levels below"* it in the case of the `enumext` environment. The *start value* can be overwritten using the `start` or `start*` keys.

These keys are intended to be used in cases where the `\resetenumext` command does not work, e.g. after an unnumbered chapter. It should preferably be set *only* on the *first level*, although it is available for all levels.

5.6.1 The command `\resetenumext`

`\resetenumext`

`\resetenumext[`⟨1⟩`]{`⟨*some counter*⟩`}` `\resetenumext[`⟨4⟩`]{`⟨*some counter*⟩`}`
`\resetenumext[`⟨2⟩`]{`⟨*some counter*⟩`}` `\resetenumext[`⟨*⟩`]{`⟨*some counter*⟩`}`
`\resetenumext[`⟨3⟩`]{`⟨*some counter*⟩`}` `\resetenumext*{`⟨*some counter*⟩`}`

The `\resetenumext` command *"resets"* the *start value* of the *"counters"* for the `enumext` and `enumext*` environments along with the *"internal counters"* used by the keys `resume` *without value* and `resume*` according to the value of {⟨*some counter*⟩}. For example `\resetenumext{chapter}` will *"reset"* the numbering of *"all levels"* of the `enumext` environment for each execution of a *"numbered"* chapter.

The *optional argument* of the form `[1]`, `[2]`, `[3]`, `[4]` *"reset"* the values for levels `1`, `2`, `3` and `4` of the `enumext` environment, the form `[*]` *"reset"* the values for the `enumext*` environment. If is run *without* the *optional argument*, it will *"reset"* the values for *"all levels"* of the `enumext` environment.

The *starred argument* '`*`' will *"reset"* the values for *"all levels"* of the `enumext` and `enumext*` environments.

5.7 Keys for `multicols`

`columns` = {⟨*integer*⟩} default: *1*

Set the *number of columns* to be used by the `multicols` environment within the environments `enumext` and `keyans`. The value must be a positive integer less than or equal to `10`. In the `enumext*` and `keyans*` environments they correspond to the default number of columns (without joining) and internally adjust the value of `\itemwidth`.

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

Set the *space between* columns used by the `multicols` environment within the environments `enumext` and `keyans`. 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. In the `enumext*` and `keyans*` environments they correspond to the *space between* columns (without joining) and internally adjust the value of `\itemwidth`.

5.8 Keys for minipage

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

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

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

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

5.8.1 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"*.

5.8.2 The key mini-right

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

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

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

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

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

6 The storage system

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

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

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

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

For security reasons the *optional argument* of the inner levels or the nested environment are *filtered* by excluding all ⟨keys⟩ related to the *"storage system"* (§6.1) along with the keys mini-env, mini-sep, mini-right, mini-right*, series, resume and resume* when storing in *sequence* {⟨store name⟩} set by save-ans key.

6.1 Keys for storage system

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

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

Sets the *name* of the *sequence* and *prop list* in which the {⟨contents⟩} will be *"stored"* by \anskey and anskey* in enumext and enumext* environments and the current ⟨labels⟩ for \item* and \anspic* in the environments keyans, keyans* and keyanspic. If the *sequence* or *prop list* {⟨store name⟩} 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 argument* 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 structure"* 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 *"stored keys"* by default in the *sequence* {⟨*store name*⟩} would be `nosep, columns=2`, but using the key `save-key={columns=3}` will overwrite and the *"stored key"* in the *sequence* {⟨*store name*⟩} are only `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 environments `keyans`, `keyans*` and `keyanspic` and storing them in the *sequence* and *prop list* {⟨*store name*⟩} set by `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.

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

This is a *"switch-key"* that does not receive an argument and disables the *"storing content"* in the *sequence* and *prop list* {⟨*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` command or use `anskey*` environment and *"without"* interfering with the `check-ans` key.

6.1.1 Keys for `label` and `ref`

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

Activates the *"internal label and ref"* mechanism for referencing *"stored content"* in *prop list* {⟨*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 *prop list* {⟨*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` in *prop list* `test` within the environment in which the key `save-ans=test` was set.

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

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

6.1.2 Keys for `wrap` and `marks`

The `enumext` package provides a set of ⟨*keys*⟩ to set and manipulate *"symbol marks"* associated with *"answers"* and how they are displayed and stored in the *sequence* and *prop list*.

The ⟨*keys*⟩ available for the `\anskey` command and the `anskey*` environment can be passed *"only"* in the *optional argument* in the *"first level"* of the `enumext` or `enumext*` environment.

The ⟨*keys*⟩ available for the `keyans` and `keyans*` environments can be passed locally in the *optional argument*, at the *"first level"* of the `enumext` or `enumext*` environment or via the `\setenumext` command with one minor difference, when ⟨*keys*⟩ are passed through the *"first level"* of the `enumext` or `enumext*` environment they are set in *"both"* environments, but when they are passed using the `\setenumext` command they are set *"individually"* in each environment.

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

Display the *symbol* set by the `mark-ans` key to the left of the *mandatory argument* ⟨*content*⟩ passed to the `\anskey` command and ⟨*body*⟩ for the `anskey*` environment using the `wrap-ans` key if set.
For `\item*` and `\anspic*` the `keyans`, `keyans*` and `keyanspic` environments it will display the *symbol* set by the `mark-ans*` key to the left of the current ⟨*label*⟩ and *optional argument*. If the *optional argument* is present in `\item*` or `\anspic*` it will be shown using `wrap-opt` key.

Keys for `\anskey` and `anskey*`

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

Sets the *symbol* to be displayed in the left margin for `\anskey` command and `anskey*` environment when using the key `show-ans`. The *"symbol"* is placed in a box of width equal to the value of `labelwidth` at the current level, separated by the value of the key `mark-sep` and aligned by the value of the key `mark-pos`. This key is not affected by the keys `font` or `wrap-label` so if you want to apply *styling* you have to do it directly, for example: `mark-ans={\textcolor{red}{\textbf{\textasteriskcentered}}}`

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

Sets the *aligned* of the *"symbol"* defined by `mark-ans` key for `\anskey` command and `anskey*` environment. 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 `mark-sep` key.

mark-sep = {⟨rigid length⟩} default: *labelsep*

Sets the *horizontal space* between the box containing the *"symbol"* defined by mark-ans key and the *mandatory argument* ⟨content⟩ passed to the \anskey command and the *body* in anskey* environment.

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

Wraps the *mandatory argument* ⟨content⟩ 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 how the *argument* or *body* is displayed 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}'.

Keys for keyans, keyans* and keyanspic

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

Sets the *symbol* to be displayed in the left margin for \item* and \anspic* for the keyans, keyans* and keyanspic environments when using the key show-ans. The *"symbol"* is placed in a box of width equal to the value of labelwidth of the environment in which it is executed, separated by the value of the key mark-sep* and aligned by the value of the key mark-pos*. This key is not affected by the keys font or wrap-label so if you want to apply *styling* you have to do it directly, for example: mark-ans*={\textcolor{red}{\textbf{\textasteriskcentered}}}.

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

Sets the *aligned* of the *"symbol"* defined by mark-ans* key for the keyans, keyans* and keyanspic environments. The *"symbol"* is aligned in a box with the same dimensions of the label box defined by labelwidth key of the environment in which it is executed and separated by the value of the mark-sep* key.

mark-sep* = {⟨rigid length⟩} default: *labelsep*

Sets the *horizontal space* between the box containing the *"symbol"* defined by mark-ans* key and the current ⟨label⟩ for \item* and \anspic* in the keyans, keyans* and keyanspic environments.

wrap-ans* = {⟨code {#1} more code⟩} default: *not used*

Wraps the *current* ⟨label⟩ when using the show-ans key for \item* and \anspic* referenced by {#1} in the keyans, keyans* and keyanspic environments after executing the align and font keys. The {⟨code⟩} must be passed between braces and *only* affects how the ⟨label⟩ is displayed and NOT the *"stored label"* in the *sequence* and *prop list* {⟨store name⟩} set by save-ans key. This key overwrites the key wrap-label and if is passed using \setenumext it is necessary to use double '{##1}'. For example, if you want the ⟨label⟩ to be displayed in red when using show-ans you just set wrap-ans*={\textcolor{red}{#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}'.

6.1.3 Keys for debug and checking

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

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

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

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

6.2 The command \anskey

\anskey `\anskey[⟨keys⟩]{⟨content⟩}`

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

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

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

6.2.1 Keys for \anskey

By default the *mandatory argument* ⟨*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: *not set*

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

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

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

Example

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

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

6.3 The environment anskey*

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

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

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

The anskey* environment is implemented using the new *"collect code"* c-type argument part of LaTeX release 2025-06-01[13]. \begin{anskey*} and \end{anskey*} must be in different lines and should not appear within verbatim environments or commands. 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, LaTeX will return an "error" message if this happens. In a similar way comments "%" or "any character" after \end{anskey*} on the same line LaTeX will return a "warning" message.

6.3.1 Keys for anskey*

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

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

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

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

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

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

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

Example

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

  \item Text containing our instructions or questions.
    \begin{enumext}
      \item Question.
        \begin{anskey*}
          ⟨second answer⟩
        \end{anskey*}
    \end{enumext}

  \item Text containing our instructions or questions.
    \begin{anskey*}
      ⟨third answer⟩
    \end{anskey*}

  \item Text containing our instructions or questions.
    \begin{anskey*}
      ⟨fourth answer⟩
    \end{anskey*}
\end{enumext}
```

⋆ 5. Text containing our instructions or questions.

[5] First answer with `verbatim`

6. Text containing our instructions or questions.

   (a) Question.

   [6] second answer

7. Text containing our instructions or questions.

[7] third answer

8. Text containing our instructions or questions.

[8] fourth answer

6.4 The environments keyans and keyans*

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

The `keyans` and `keyans*` environments are *"enumerated list"* environments designed for *"multiple choice"* questions activated by the `save-ans` key.

This environments can NOT be nested and must always be at the *"first level"* of the `enumext` environment, the commands `\item` and `\item[⟨custom⟩]` work in the usual and the command `\item(⟨columns⟩)` is available for the `keyans*` environment.

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

```
\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*)`.

The keys `mark-ans*`, `mark-pos*`, `mark-sep*`, `save-sep`, `wrap-opt`, `wrap-ans*`, `show-ans` and `show-pos` are available for both environments.

6.4.1 The \item* in keyans and keyans*

\item* `\item*`
`\item*[⟨content⟩]`

The `\item*` and `\item*[⟨content⟩]` command *"store"* the current ⟨*label*⟩ set by `label` key next to the *optional argument* ⟨*content*⟩ 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.

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

Some text

6.5 The environment keyanspic

keyanspic `\begin{keyanspic}[⟨key = val⟩] \anspic*[⟨content⟩]{⟨drawing or tabular⟩} \end{keyanspic}`

The `keyanspic` environment is an *"enumerated list"* environment activated by the `save-ans` key that has the same configuration for *"spacing"* and ⟨*label*⟩ as the `keyans` environment that uses the `\anspic` command instead of `\item`. It is intended for placing *drawings or tabular* with ⟨*label*⟩ centered *above* or *below* in a *single line* or *upper and lower* layout style.

When the `keyanspic` environment is used *without keys* the ⟨*labels*⟩ are centered *below* the *drawings or tabular* in a *single line* layout style.
A representation of the output can be seen in the figure 6.

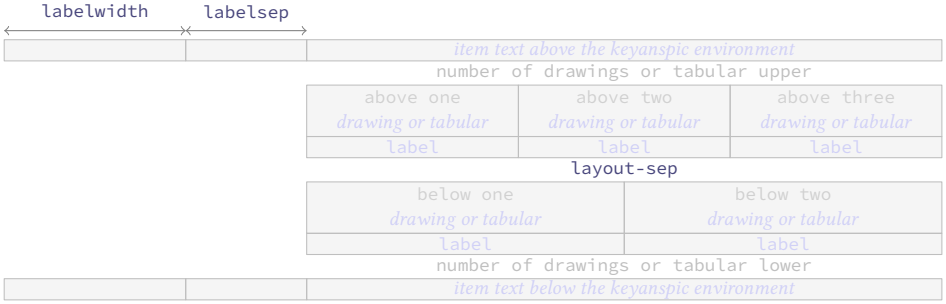


Figure 6: Representation of the `keyanspic` environment with `layout-sty={`⟨*3, 2*⟩`}` in enumext.

This environment cannot be nested and must *always* be at the *"first level"* of the `enumext` environment, the `\item` command is disabled and ⟨*keys*⟩ cannot be set using `\setenumext`.

6.5.1 Keys for keyanspic

label-pos = {⟨*above* | *below*⟩} default: *below*

Set the *position* of ⟨*label*⟩ to be centered "above" or "below" *drawings* or *tabular* when the \anspic command is executed.

label-sep = {⟨*rubber length* | *rigid length*⟩} default: *internal adjustment*

Set the *vertical spacing* between the ⟨*label*⟩ centered "above" or "below" and *drawings* or *tabular* when running the \anspic command.

layout-sty = {⟨*n° upper* , *n° lower*⟩} default: *not set*

Set the *number* of *drawings* or *tabular* that will be distributed "upper" and "lower" within the environment when executing the \anspic command. The value must be passed in braces and if not set or the ⟨*n° lower*⟩ is omitted the *drawings* or *tabular* will be put on a *single line*.

layout-sep = {⟨*rubber length* | *rigid length*⟩} default: *adjusted parsep from keyans*

Set the *vertical separation* between the number of *drawings* or *tabular* placed at the "upper" and "lower" within the environment when executing the \anspic command. Internally adjusts the parsep value taken from the keyans environment.

layout-top = {⟨*rubber length* | *rigid length*⟩} default: *adjusted topsep from keyans*

Set the *vertical space* added to both the top and bottom of the environment. Internally adjust the value of topsep taken from keyans environment.

The keys mark-ans*, mark-pos*, mark-sep*, save-sep, wrap-opt, wrap-ans*, show-ans and show-pos are available for this environment.

6.5.2 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 *optional argument* ⟨*content*⟩ 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=true,nosep]
  \item Question with images and labels below.

    \begin{keyanspic}[layout-sty={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}

  \item Question with images and labels above.

    \begin{keyanspic}[label-pos=above, layout-sty={3,2},layout-sep=0.25cm]
      \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}

  \item Question with images and labels below on a single line.

    \begin{keyanspic}
      \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 and labels below.

A)

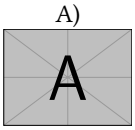
B)

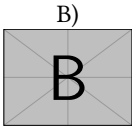
C)

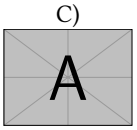
D)

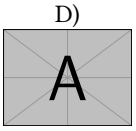
∗ E) [note]

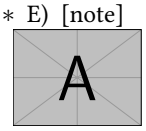
2. Question with images and labels above.

A)


B)


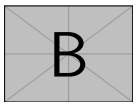
C)


D)


∗ E) [note]


3. Question with images and labels below on a single line.

A)



B)

C)

D)

∗ E) [note]

Remember to pass the `alt={`*description*`}` key to the `\includegraphics` command when creating a *tagged* PDF.

6.6 Printing stored content

6.6.1 The command `\getkeyans`

`\getkeyans` `\getkeyans{`*store name* `:` *position*`}`

The command `\getkeyans` prints the *"stored content"* in *prop list* `{`*store name*`}` defined by `save-ans` key in the ⟨*position*⟩ returned by the `show-pos` key.

The *"stored content"* can only be accessed *after* it is stored, if `{`*store name*`}` does not exist the command will return an error.

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

6.6.2 The command `\foreachkeyans`

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

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

Options for command

`sep` = `{`*code*`}` default: *{;}*

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

`step` = `{`*integer*`}` default: `1`

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

`start` = `{`*integer*`}` default: `1`

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

`stop` = `{`*integer*`}` default: `0`

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

before = {⟨code⟩} default: *empty*

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

after = {⟨code⟩} default: *empty*

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

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

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

6.6.3 The command \printkeyans

\printkeyans

\printkeyans{⟨store name⟩}
\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 or 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=1,show-pos=true,nosep,save-ref=true]
  \item Factor $3x+3y+3z$. \anskey{$3(x+y+z)$}
  \item True False

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

  \item Related to Linux

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

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

\printkeyans{sample}
```

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

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

2. True False
   (a) LaTeX2e is cool?
   [2] Very True!

3. Related to Linux
   (a) You use linux?
   [3] Yes
   (b) Rate the following package and class
      i. `xsim`
      [4] very good
      ii. `exsheets`
      [5] obsolete

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

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

7 Full examples

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

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

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

Example 1

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

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

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

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

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

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

Example 2

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

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

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 ※

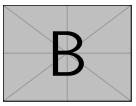
Example 3

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 right side:
   - (A) value
   - (B) value
   - (C) value
   - (D) correct
   - (E) value



Test keys

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

Example 4

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

Factor $x^2 - 2x + 1$

Factor $3x + 3y + 3z$

The following questions need to be cuaqtified :)

True False

(a) $\alpha > \delta$

(b) LaTeX2e is cool?

Related to Linux

(a) You use linux?

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

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

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

Example 5

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

1

Which choice best describes what happens in the passage?

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

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)

Example 6

Adapted from the response to Environment for enumerate environment.

8.5a, KSC **10.** sample

A sample
✓ **B** answer
C sample
D sample

9.5a, KSC **11.** sample

A sample
B sample
C sample
✓ **D** answer

12. sample

A sample
B answer
C sample
D sample

13. sample

A sample
B sample
C sample
D answer

10. B (8.5a, KSC)
11. D (9.5a, KSC)
12. B (10.5a, KSC)
13. D (11.5a, KSC)

8 Tagged PDF examples

This section is just to show the compatibility of enumext with *tagged* PDF using `lualatex`. The attached files here are just for testing and are intended as examples and, in a way, to simplify the time of Matthew Bertucci (`@mbertucci`) when he sees this excellent package and adds it to The LaTeX Tagged PDF repository.

To compile the tests with `lualatex-dev` the packages `multicol`, `unicode-math`, `geometry`, `graphicx`, `luamml` and `hyperref` are required along with the line:

```
\DocumentMetadata{lang = en-US, pdfversion = 2.0, pdfstandard = ua-2, tagging=on,}
```

All examples have been checked using veraPDF together with ngpdf.

- The file `enumext-01.tex` contains the basic tests for the `enumext` and `enumext*` environments and the nesting between them plus the use of the `label`, `labelwidth`, `labelsep`, `ref`, `align` and `wrap-label` keys. Source file and *tagged* PDF.
- The file `enumext-02.tex` contains the tests for the `enumext` and `enumext*` environments and the support for `minipage` and `multicols` environments using the keys `columns`, `columns-sep`, `mini-env`, `mini-right` and `\miniright` command. Source file and *tagged* PDF.
- The file `enumext-03.tex` contains the tests for the `enumext` and `keyanspic` environments activated by the `save-ans` key together with the `save-sep` and `save-ref` keys and the `\printkeyans` command. Source file and *tagged* PDF.
- The file `enumext-04.tex` contains the tests for the `\anskey` command and the `anskey*` environment activated by the `save-ans` key along with the `\getkeyans` and `\printkeyans` commands. Source file and *tagged* PDF.
- The file `enumext-05.tex` contains the tests for the environments `keyans`, `keyans*` and `keyanspic` activated by the key `save-ans` together with the keys `no-store` and `show-ans` and the commands `\setenumext`, `\setenumextmeta`, `\printkeyans` and `\foreachkeyans`. Source file and *tagged* PDF.
- The file `enumext-06.tex` contains the tests for the environments `enumext` and `enumext*` for *fake* `itemize` and `description`. Source file and *tagged* PDF.
- The file `enumext-07.tex` contains the tests for starting the environments with `\setenumext{resume}`, the `\resetenumext` command and the `series`, `resume`, `resume*` and `reset` keys. Source file and *tagged* PDF.

9 The way of non-enumerated lists

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

When *tagged* PDF is active the default `description` style is NOT available due to the redefinition of `\makelabel` for the `align` key which uses `\makebox` in this case, meaning that `\item[`⟨*content*⟩`]` will not extend beyond `\labelwidth` which causes overlaps,

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` and `wrap-label*` keys 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 }
  {%
    \SuspendTagging{\parbox}%
    \IfBooleanTF{#1}
      {\strut\smash{\parbox[t]{\labelwidth}{\raggedright{#2}}}}%
      {\strut\smash{\parbox[t]{\labelwidth}{\raggedleft{#2}}}}%
    \ResumeTagging{\parbox}%
  }
```

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.

Why has it taken so long?

One of the setbacks, beyond my laziness, was including compatibility with *tagged* PDF. To be honest, it's something I never considered at any point, but I firmly believe that being able to create *accessible documents* provides a great opportunity in the world of mathematics education. From my perspective as a *high school* teacher, beyond theorems and deep mathematics, the use of exercise lists is one of the most common things. Being able to open the way to work in parallel with those who have different abilities is really important and I regret not having looked into this in the past. I hope that enumext serves this purpose and inspires more users and authors to follow this path.

10 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, 2025.

[4] González, Pablo. "scontents - Stores LaTeX contents in memory or files". Available from CTAN, https://www.ctan.org/pkg/scontents, 2025.

[5] The LaTeX Project. "enumerate – Enumerate with redefinable labels". Available from CTAN, https://www.ctan.org/pkg/enumerate, 2025.

[6] Bezos, Javier. "Customizing lists with the enumitem package". Available from CTAN, https://www.ctan.org/pkg/enumitem, 2025.

[7] Berry, Karl. "LaTeX 2_ε: An Unofficial Reference Manual". Available from CTAN, https://ctan.org/pkg/latex2e-help-texinfo, 2025.

[8] The LaTeX Project. "Extensive support for hypertext in LaTeX". Available from CTAN, https://www.ctan.org/pkg/hyperref, 2025.

[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, 2025.

[11] The LaTeX Project. “The LaTeX3 Interfaces”. Available from CTAN, https://www.ctan.org/pkg/l3kernel, 2025.

[12] The LaTeX Project. “The LaTeX 2_ε sources”. Available from CTAN, https://ctan.org/tex-archive/macros/latex/base, 2025.

[13] The LaTeX Project. “LaTeX News, Issue 41, June 2025”. Available from CTAN, https://ctan.org/tex-archive/macros/latex/base, 2025.

[14] The LaTeX Project. “LaTeX for authors current version”. Available from CTAN, https://ctan.org/pkg/latex-base, 2025.

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

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

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

[18] Fischer, Ulrike. “tagpdf – LaTeX kernel code for PDF tagging”. Available from CTAN, https://www.ctan.org/pkg/tagpdf, 2025.

[19] The LaTeX Project. “latex-lab – LaTeX laboratory”. Available from CTAN, https://www.ctan.org/pkg/latex-lab, 2025.

[20] Mittelbach, Frank. “LaTeX’s socket management”. Available from CTAN, https://ctan.org/tex-archive/macros/latex/base, 2025.

11 Change history

v1.7 (ctan), 2025-07-10
- Fixed `\setenumext{enumext*}{resume}`.
- Fixed bad interaction between `\setenumext` and the `resume` key.
- The behavior of the key `resume*` has been updated and documented.

v1.6 (ctan), 2025-07-04
- Syntax simplification for `\setenumextmeta`.
- Environments can be started with the key `resume` *without value*.
- Add `\resetenumext`, `reset` and `reset*` keys.
- The `resume`, `resume*` and `series` keys can now be set per level.
- Fixed bad interaction between `\printkeyans` and the `resume`, `resume*` keys.

v1.5 (ctan), 2025-06-11
- Replacing `\regex_match:` (deprecated) with `\regex_if_match:`.
- Add keys `beginpenalty`, `midpenalty` and `endpenalty`.

v1.4 (ctan), 2025-06-09
- Improved implementation of the `start` key for *tagged* PDF.
- Improved implementation of the `ref` key.
- Fixed the behavior of the `save-sep` key.
- Fixed the behavior of the `resume*` key.

v1.3 (ctan), 2025-06-01
- Removed dependency on the `scontents` package.
- The `anskey*` environment has been rewritten using the new `c-type` argument.

v1.2 (ctan), 2025-03-28
- Replace signature (prevent expansion for optional argument).
- Solve Inconsistent local/global assignment.

v1.1 (ctan), 2024-11-14
- Fixed implementation for `font` and `base-fix` keys.
- Added new keys for symbol marks.
- Update and improvements in the internal code.
- Adjustments in the documentation.

v1.0 (ctan), 2024-11-01
- First public release.

12 Index of Documentation

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

L

Labels provide by enumext:

P

Packages:

R

T

13 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.

13.1 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.

13.2 Initial set up

Start the DocStrip guards.

```
⟨*package⟩
```

Identify the internal prefix (LaTeX3 DocStrip convention) for l3doc class.

```
⟨@@=enumext⟩
```

13.3 Declaration of the package

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

```
\NeedsTeXFormat{LaTeX2e}[2025-06-01]
```

Now declare the enumext package.

```
\ProvidesExplPackage {enumext} {2025-07-10} {1.7} {Enumerate exercise sheets}
```

Finally check if the multicol package 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}[2025-05-25]
      }
  }
```

13.4 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, anskey* environment 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_keyans_env_bool
\g__enumext_start_line_tl
\g__enumext_envir_name_tl
\l__enumext_envir_name_tl

Internal variables used by functions `\__enumext_is_not_nested:`, `\__enumext_is_on_first_level:` and `\__enumext_keyans_name_and_start:` (§13.5.1).

```
\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_keyans_env_bool
\tl_new:N   \g__enumext_start_line_tl
\tl_new:N   \g__enumext_envir_name_tl
\tl_new:N   \l__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_counter:Nn` (§13.11) and then modified by the function `\__enumext_label_style:Nnn` used by `label` key (§13.14).

```
\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.*)

\l__enumext_ref_key_arg_tl
\l__enumext_ref_the_count_tl
\l__enumext_renew_counter_X_tl
\l__enumext_the_counter_X_tl

Internal variables used by `ref` key (§13.14).

```
\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_counter_#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* `\l__enumext_ref_key_arg_tl` *and others.*)

\l__enumext_series_name_tl
\l__enumext_resume_count_bool
\l__enumext_resume_count_X_bool
\l__enumext_resume_series_X_bool
\l__enumext_resume_star_key_X_bool
\g__enumext_resume_last_keys_X_tl

Internal variables used by `resume`, `resume*` and `series` keys (§13.26).

```
\tl_new:N \l__enumext_series_name_tl
\bool_new:N \l__enumext_resume_count_bool
\cs_set_protected:Npn \__enumext_tmp:n #1
  {
    \bool_new:c { l__enumext_resume_count_#1_bool    }
    \bool_new:c { l__enumext_resume_series_#1_bool   }
    \bool_new:c { l__enumext_resume_star_key_#1_bool }
    \tl_new:c   { g__enumext_resume_last_keys_#1_tl  }
  }
\clist_map_inline:nn { i, ii, iii, iv, vii } { \__enumext_tmp:n {#1} }
```

(*End of definition for* `\l__enumext_series_name_tl` *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` (§13.15) and `label` (§13.13) 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 (§13.19). The variables \l__enumext_leftmargin_X_dim and \l__enumext_itemindent_X_dim are used and set by the function __enumext_calc_hspace:NNNNNNNNNNNN (§13.41.1).

```
\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
\g__enumext_multicols_right_X_skip
\l__enumext_align_label_pos_X_str

Internal variables used by columns key (§13.23) and align key (§13.13).

```
\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 }
    \skip_new:c  { g__enumext_multicols_right_#1_skip }
    \str_new:c   { l__enumext_align_label_pos_#1_str  }
  }
\clist_map_inline:nn { i, ii, iii, iv, v } { \__enumext_tmp:n {#1} }
```

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

\g__enumext_minipage_stat_int
\l__enumext_minipage_temp_skip
\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 (§13.24.4) and the keys mini-right, mini-right*, mini-env and mini-sep (§13.22, §13.24).

```
\int_new:N  \g__enumext_minipage_stat_int
\skip_new:N \l__enumext_minipage_temp_skip
\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 bool vars \l__enumext_wrap_label_X_bool and \l__enumext_wrap_label_opt_X_bool are used by wrap-label and wrap-label* keys (§13.13), the integer \l__enumext_start_X_int are used by the start and start* keys (§13.15), the token list \l__enumext_fake_item_indent_X_tl is used by itemindent key (§13.19.1), the variables \l__enumext_label_fill_left_X_tl and \l__enumext_label_fill_left_X_tl are used by the align key (§13.13). 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 (§13.21).

```
\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_current_label_tl
\l__enumext_store_current_opt_arg_tl

The variable \l__enumext_store_active_bool setting by save-ans key (§13.29.1) 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 variables \l__enumext_store_current_label_tl and \l__enumext_store_current_opt_arg_tl save the *current label* and *optional argument* of \item* (§13.40) and \anspic* (§13.45.2) for the keyans, keyans* and keyanspic environments.

```
\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_current_label_tl
\tl_new:N   \l__enumext_store_current_opt_arg_tl
```

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

\l__enumext_store_anskey_arg_tl
\l__enumext_store_anskey_env_tl
\l__enumext_write_anskey_env_bool
\l__enumext_write_anskey_env_file_name_tl
\l__enumext_write_anskey_env_file_iow

The variable \l__enumext_store_anskey_arg_tl save the *argument* of \anskey (§13.33) and the variables \l__enumext_store_anskey_env_tl save the ⟨*body*⟩ of the environment anskey* (§13.34).

The variables \l__enumext_write_anskey_env_bool, \l__enumext_write_anskey_env_file_name_tl and \l__enumext_write_anskey_env_file_iow they are used by the write-env and overwrite keys in the anskey* environment implementation.

```
\tl_new:N   \l__enumext_store_anskey_arg_tl
\tl_new:N   \l__enumext_store_anskey_env_tl
\bool_new:N \l__enumext_write_anskey_env_bool
\tl_new:N   \l__enumext_write_anskey_env_file_name_tl
\iow_new:N  \l__enumext_write_anskey_env_file_iow
```

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

\c__enumext_anskey_env_hidden_space_str

The \c__enumext_anskey_env_hidden_space_str is a constant *string* to used to hide the ⟨*forced space*⟩ added by TEX when recording content in a macro. This *string* contains the *reserved phrase* "%^^Aenumextheol%" which is added to the end of the argument stored in *sequence* and *prop list* when the key force-eol is false.

```
\str_const:Ne \c__enumext_anskey_env_hidden_space_str
  { \c_percent_str \c_circumflex_str \c_circumflex_str A enumextheol \c_percent_str }
```

(*End of definition for* \c__enumext_anskey_env_hidden_space_str.)

\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 (§13.51).

```
\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_meta_path_str
\l__enumext_foreach_print_seq
\l__enumext_foreach_name_prop_tl
\l__enumext_foreach_default_keys_tl

Internal variables used by the \setenumextmeta command (§13.52) and \foreachkeyans command (§13.53).

```
\str_new:N \l__enumext_meta_path_str
\seq_new:N \l__enumext_foreach_print_seq
\tl_new:N  \l__enumext_foreach_name_prop_tl
\tl_new:N  \l__enumext_foreach_default_keys_tl
```

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

\l__enumext_print_keyans_starred_tl
\l__enumext_print_keyans_star_bool
\l__enumext_print_keyans_cmd_bool
\l__enumext_mark_position_str
\l__enumext_mark_position_v_str
\l__enumext_mark_position_viii_str
\l__enumext_mark_sep_tmpa_dim
\l__enumext_mark_sep_tmpb_dim
\l__enumext_show_pos_tmp_int
\g__enumext_item_symbol_aux_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 (§13.50), show-pos, show-ans, mark-pos, mark-sep keys (§13.30), item-sym* key (§13.38), save-key key (§13.30.3) and *"storing structure"*.

```
\tl_new:N   \l__enumext_print_keyans_starred_tl
\bool_new:N \l__enumext_print_keyans_star_bool
\bool_new:N \l__enumext_print_keyans_cmd_bool
\str_new:N  \l__enumext_mark_position_str
\str_new:N  \l__enumext_mark_position_v_str
\str_new:N  \l__enumext_mark_position_viii_str
\dim_new:N  \l__enumext_mark_sep_tmpa_dim
\dim_new:N  \l__enumext_mark_sep_tmpb_dim
\int_new:N  \l__enumext_show_pos_tmp_int
\tl_new:N   \g__enumext_item_symbol_aux_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_anspic_args_seq` `\l__enumext_anspic_mini_width_dim` `\l__enumext_anspic_above_int` `\l__enumext_anspic_below_int` `\l__enumext_anspic_label_above_bool` `\l__enumext_anspic_mini_pos_str` `\l__enumext_anspic_label_box` `\l__enumext_anspic_body_box` `\l__enumext_anspic_label_htdp_dim` `\l__enumext_anspic_body_htdp_dim`

Internal variables used by `keyanspic` environment and `\anspic` command (§13.45.1).

```
\seq_new:N  \l__enumext_anspic_args_seq
\dim_new:N  \l__enumext_anspic_mini_width_dim
\int_new:N  \l__enumext_anspic_above_int
\int_new:N  \l__enumext_anspic_below_int
\bool_new:N \l__enumext_anspic_label_above_bool
\str_new:N  \l__enumext_anspic_mini_pos_str
\box_new:N  \l__enumext_anspic_label_box
\box_new:N  \l__enumext_anspic_body_box
\dim_new:N  \l__enumext_anspic_label_htdp_dim
\dim_new:N  \l__enumext_anspic_body_htdp_dim
```

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

`\l__enumext_check_answers_bool` `\g__enumext_check_ans_key_bool` `\l__enumext_check_start_line_env_tl` `\l__enumext_item_wrap_key_bool` `\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 (§13.29.3) used by the `check-ans`, `no-store`, `wrap-ans*` 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
\bool_new:N \l__enumext_item_wrap_key_bool
\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 (§13.7). 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 (§13.30). 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` (§13.7) 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_standar_int` `\g__enumext_footnote_starred_int` `\g__enumext_footnote_standar_arg_seq` `\g__enumext_footnote_starred_arg_seq` `\g__enumext_footnote_standar_int_seq` `\g__enumext_footnote_starred_int_seq`

Internal variables used for redefinition of `\footnote` (§13.8).

```
\int_new:N \g__enumext_footnote_standar_int
\int_new:N \g__enumext_footnote_starred_int
\seq_new:N \g__enumext_footnote_standar_arg_seq
\seq_new:N \g__enumext_footnote_starred_arg_seq
\seq_new:N \g__enumext_footnote_standar_int_seq
\seq_new:N \g__enumext_footnote_starred_int_seq
```

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

\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`.)

13.5 Some utility functions

\keys_precompile:neN
\seq_use:NV

Non-standard kernel variants used by the `\printkeyans` command (§13.50) and `\foreachkeyans` command (§13.53).

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

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

__enumext_scan_tokens:n

The functions `\tl_rescan:nn` and `\tl_set_rescan:Nnn` provided by expl3 doesn't fit the needs of this package because it does not allow catcode changes inside the argument, so verbatim stuff used inside one of anskey* environment will not work. Here we create a private copy of `\tex_scantokens:D` which will serve our purposes. See the answer by Ulrich Diez in How do use {<setup>} in \tl_set_rescan:Nnn to replace \scantokens?.

```
\cs_new_protected:Npn \__enumext_scan_tokens:n #1 { \tex_scantokens:D {#1} }
```

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

__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_if_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_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`.)

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

```
\cs_new_protected:Nn \__enumext_unskip_unkern:
  {
    \int_case:nnT { \lastnodetype }
      {
        { 11 }{ \unskip }
        { 12 }{ \unkern }
      }
  }
```

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

13.5.1 Utilities for environments and levels

`\__enumext_is_not_nested:` `\__enumext_is_on_first_level:` The function `\__enumext_is_not_nested:` set the variables `\g__enumext_standar_bool` and `\g__enumext_starred_bool` to *"true"* only if the environments `enumext` and `enumext*` are NOT nested in each other and save the environment name in `\l__enumext_envir_name_tl`.

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

```
260               {
261                 \bool_gset_true:N \g__enumext_starred_bool
262               }
263           }
264       }
265   }
```

The function `\__enumext_is_on_first_level:` will set the variables `\l__enumext_standar_first_bool` (§13.29.1), `\l__enumext_starred_first_bool` (§13.29.1) 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.

```
266 \cs_new_protected:Nn \__enumext_is_on_first_level:
267   {
268     \bool_lazy_all:nT
269       {
270         { \bool_if_p:N \g__enumext_standar_bool }
271         { \int_compare_p:nNn { \l__enumext_level_int } = { 1 } }
272         { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
273       }
274       {
275         \bool_set_true:N \l__enumext_standar_first_bool
276         \tl_gset:Nn \g__enumext_envir_name_tl { enumext }
277         \tl_gset:Ne \g__enumext_start_line_tl
278           {
279             on~line~\exp_not:V \inputlineno
280           }
281       }
282     \bool_lazy_all:nT
283       {
284         { \bool_if_p:N \g__enumext_starred_bool }
285         { \int_compare_p:nNn { \l__enumext_level_h_int } = { 1 } }
286         { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
287       }
288       {
289         \bool_set_true:N \l__enumext_starred_first_bool
290         \tl_gset:Nn \g__enumext_envir_name_tl { enumext* }
291         \tl_gset:Ne \g__enumext_start_line_tl
292           {
293             on~line~\exp_not:V \inputlineno
294           }
295       }
296   }
```

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

`\__enumext_keyans_name_and_start:`

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

```
297 \cs_new_protected:Nn \__enumext_keyans_name_and_start:
298   {
299     \str_case:en { \@currenvir }
300       {
301         {keyans}
302           {
303             \tl_set:Nn \l__enumext_envir_name_tl { keyans }
304             \tl_set:Ne \l__enumext_check_start_line_env_tl
305               {
306                 in~'keyans'~start~on~line~\exp_not:V \inputlineno
307               }
308           }
309         {keyans*}
310           {
311             \tl_set:Nn \l__enumext_envir_name_tl { keyans* }
312             \tl_set:Ne \l__enumext_check_start_line_env_tl
313               {
314                 in~'keyans*'~start~on~line~\exp_not:V \inputlineno
315               }
316           }
317         {keyanspic}
318           {
```

```
            \tl_set:Nn \l__enumext_envir_name_tl { 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_name_and_start:`.)

13.5.2 Utilities for log and terminal

`\__enumext_reset_global_vars:`
`\__enumext_reset_global_int:`
`\__enumext_reset_global_bool:`
`\__enumext_reset_global_tl:`

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

```
\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:`.)

13.6 Copying `list` and `minipage` environments

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

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

And `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 `list` environment or a related command.

For compatibility with *tagged* PDF we should use `\NewCommandCopy` and not `\cs_new_eq:NN` for `\item`. When *tagged* PDF is active `\item` is redefined using `ltcmd` (see `latex-lab-block`[19]).

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

The functions `\__enumext_start_list:nn` and `\__enumext_stop_list:` correspond to copies of `\list` and `\endlist` from plain definition of `list` environment, the function `\__enumext_item_std:w` is a copy of the `\item` command.

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

The functions `\__enumext_minipage:w` and `\__enumext_endminipage:` correspond to copies of `\minipage` and `\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_start_list:nn` *and others.*)

13.7 Compatibility with hyperref and footnotehyper

`\__enumext_after_hyperref:`
`\__enumext_hypertarget:nn`
`\__enumext_phantomsection:`

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 }
```

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:
  {
    \IfPackageLoadedT { hyperref }
      {
        \msg_info:nnn { enumext } { package-load } { hyperref }
        \bool_set_true:N \l__enumext_hyperref_bool
        \IfHyperBoolean{hyperfootnotes}
          {
            \bool_set_true:N \l__enumext_footnotes_key_bool
          }
          {  }
      }
```

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 }
          }
          {
            \bool_set_false:N \l__enumext_footnotes_key_bool
```

```
399         }
400     }
```

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.

```
401     \bool_if:NTF \l__enumext_hyperref_bool
402       {
403         \cs_new_eq:NN \__enumext_hypertarget:nn \hypertarget
404         \cs_new_eq:NN \__enumext_phantomsection: \phantomsection
405       }
406       {
407         \cs_new_eq:NN \__enumext_hypertarget:nn \use_none:nn
408         \cs_new_eq:NN \__enumext_phantomsection: \prg_do_nothing:
409       }
410   }
```

(*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.

```
411 \cs_new_protected:Npn \__enumext_newlabel:nn #1 #2
412   {
413     \protected@write \@auxout { }
414       {
415         \token_to_str:N \newlabel {#1}
416           {
417             {#2}
418             \bool_if:NT \l__enumext_hyperref_bool
419               { { \thepage } {#2} {#1} }
420             { }
421           }
422       }
423     \__enumext_hypertarget:nn {#1} { }
424     \__enumext_phantomsection:
425   }
```

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

13.8 Internal redefining `\footnote` command

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

`\__enumext_footnotetext:nn`
`\__enumext_renew_footnote:`
`\__enumext_print_footnote:`
`\__enumext_renew_footnote_mini:`
`\__enumext_print_footnote_mini:`

Redefinition of the `\footnote` command using `\footnotetext` and `\footnotemark` for the `mini-env` key in the `enumext` and `keyans` environments.

```
426 \cs_new_protected:Nn \__enumext_footnotetext:nn
427   {
428     \footnotetext[#1]{#2}
429   }
430 \cs_new_protected:Nn \__enumext_renew_footnote:
431   {
432     \RenewDocumentCommand \footnote { o +m }
433       {
434         \tl_if_novalue:nTF {##1}
435           {
436             \stepcounter{footnote}
437             \int_gset_eq:Nc \g__enumext_footnote_standar_int { c@footnote }
438           }
439           {
440             \int_gset:Nn \g__enumext_footnote_standar_int { ##1 }
441           }
442         \footnotemark [ \g__enumext_footnote_standar_int ]
443         \seq_gput_right:Nn \g__enumext_footnote_standar_arg_seq { ##2 }
444         \seq_gput_right:NV
445           \g__enumext_footnote_standar_int_seq \g__enumext_footnote_standar_int
```

```
      }
  }
\cs_new_protected:Nn \__enumext_print_footnote:
  {
    \seq_if_empty:NF \g__enumext_footnote_standar_int_seq
      {
        \seq_map_pairwise_function:NNN
          \g__enumext_footnote_standar_int_seq
          \g__enumext_footnote_standar_arg_seq
          \__enumext_footnotetext:nn
      }
    \seq_gclear:N \g__enumext_footnote_standar_arg_seq
    \seq_gclear:N \g__enumext_footnote_standar_int_seq
  }
```

The enumext* and keyans* environments are implemented using minipage so we must also redefine \footnote to keep these numbering as if it were part of the document.

```
\cs_new_protected:Nn \__enumext_renew_footnote_mini:
  {
    \RenewDocumentCommand \footnote { o +m }
      {
        \tl_if_novalue:nTF {##1}
          {
            \stepcounter{footnote}
            \int_gset_eq:Nc \g__enumext_footnote_starred_int { c@footnote }
          }
          {
            \int_gset:Nn \g__enumext_footnote_starred_int { ##1 }
          }
        \footnotemark [ \g__enumext_footnote_starred_int ]
        \seq_gput_right:Nn \g__enumext_footnote_starred_arg_seq { ##2 }
        \seq_gput_right:NV
          \g__enumext_footnote_starred_int_seq \g__enumext_footnote_starred_int
      }
  }
\cs_new_protected:Nn \__enumext_print_footnote_mini:
  {
    \seq_if_empty:NF \g__enumext_footnote_starred_int_seq
      {
        \seq_map_pairwise_function:NNN
          \g__enumext_footnote_starred_int_seq
          \g__enumext_footnote_starred_arg_seq
          \__enumext_footnotetext:nn
      }
    \seq_gclear:N \g__enumext_footnote_starred_arg_seq
    \seq_gclear:N \g__enumext_footnote_starred_int_seq
  }
```

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

`\__enumext_renew_footnote_standar:`
`\__enumext_print_footnote_standar:`
`\__enumext_renew_footnote_starred:`
`\__enumext_print_footnote_starred:`

We encapsulate the redefinition of \footnote to pass it to internal __enumext_mini_page environment used by the mini-env key in the enumext and keyans environments. We will run the redefinition when *tagged* PDF is active or when the footnotehyper package is not loaded.

```
\cs_new_protected:Nn \__enumext_renew_footnote_standar:
  {
    \bool_if:NT \g__enumext_standar_bool
      {
        \IfDocumentMetadataTF
          {
            \__enumext_renew_footnote:
          }
          {
            \bool_if:NF \l__enumext_footnotes_key_bool
              {
                \__enumext_renew_footnote:
              }
          }
      }
  }
\cs_new_protected:Nn \__enumext_print_footnote_standar:
  {
```

```
    \bool_if:NT \g__enumext_standar_bool
      {
        \IfDocumentMetadataTF
          {
            \__enumext_print_footnote:
          }
          {
            \bool_if:NF \l__enumext_footnotes_key_bool
              {
                \__enumext_print_footnote:
              }
          }
      }
  }
```

We encapsulate the redefinition of `\footnote` to pass it to the `enumext*` and `keyans*` environments. We will run the redefinition when *tagged* PDF is active or when the `footnotehyper` package is not loaded.

```
\cs_new_protected:Nn \__enumext_renew_footnote_starred:
  {
    \IfDocumentMetadataTF
      {
        \__enumext_renew_footnote_mini:
      }
      {
        \bool_if:NF \l__enumext_footnotes_key_bool
          {
            \__enumext_renew_footnote_mini:
          }
      }
  }
\cs_new_protected:Nn \__enumext_print_footnote_starred:
  {
    \IfDocumentMetadataTF
      {
        \__enumext_print_footnote_mini:
      }
      {
        \bool_if:NF \l__enumext_footnotes_key_bool
          {
            \__enumext_print_footnote_mini:
          }
      }
  }
```

In `enumext*` and `keyans*` environments we need to use *"hooks"* to print `\footnote` with support for *tagged* PDF.

```
\__enumext_after_env:nn { enumext* }
  {
    \__enumext_print_footnote_starred:
  }
\__enumext_after_env:nn { keyans* }
  {
    \__enumext_print_footnote_starred:
  }
```

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

13.9 The internal `minipage` environment

`\__enumext_internal_mini_page:`
`__enumext_mini_env*`

The function `\__enumext_internal_mini_page:` creates a internal `__enumext_mini_page` environment (*custom version* of `minipage`) setting the `\if@minipage` switch to *"false"* to allow spaces at the *"above"* of the environment, plus we will add `\skip_vertical:N \c_zero_skip` to maintain alignment on *"top"* in the first part and `\skip_vertical:N \c_zero_skip` in the second part to allow spaces *"below"*. This environment will be used internally by the `mini-env` key, it is NOT documented in the user interface and is for internal use only. Within this environment we redefine `\footnote` to make them look the same as if they were elsewhere in the document. This implementation is adapted from the answer given by Max Chernoff (`@MaxChernoff`) in Customize `minipage` to support spaces below it.

This function is passed to the function `\__enumext_safe_exec:` in the `enumext` environment definition (§13.42) and `\__enumext_safe_exec_vii:` in the `enumext*` environment definition (§13.47).

```
\cs_new_protected:Nn \__enumext_internal_mini_page:
  {
```

```
    \int_compare:nNnT { \l__enumext_level_int } = { 0 }
      {
        \DeclareDocumentEnvironment{__enumext_mini_page}{ m }
          {
            \__enumext_renew_footnote_standar:
            \__enumext_minipage:w [ t ] { ##1 }
              \legacy_if_gset_false:n { @minipage }
              \skip_vertical:N \c_zero_skip
          }
          {
              \skip_vertical:N \c_zero_skip
            \__enumext_endminipage:
            \__enumext_print_footnote_standar:
          }
      }
  }
```

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

13.10 Definition of public dimension

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

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

13.11 Definition of counters

`\__enumext_define_counter:Nn`
`enumXi`
`enumXii`
`enumXiii`
`enumXiv`
`enumXv`
`enumXvi`
`enumXvii`
`enumXviii`

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

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

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

`\c@__enumext_resume_i_int`
`\c@__enumext_resume_ii_int`
`\c@__enumext_resume_iii_int`
`\c@__enumext_resume_iv_int`
`\c@__enumext_resume_vii_int`

In version 1.6 the command `\resetenumext` (§13.27) was added which internally uses `\counterwithin*` so for its correct operation, we will create *"real counters"* instead of the *"integer variables"* for the keys `resume` and `resume*`.

```
\cs_set_protected:Npn \__enumext_tmp:n #1
  {
    \cs_if_exist:cTF { c@ __enumext_resume_#1_int }
      { \msg_fatal:nne { enumext } { counters }{ __enumext_resume_#1_int } }
      {
        \newcounter { __enumext_resume_#1_int }
      }
  }
\clist_map_inline:nn {i,ii,iii,iv,vii} { \__enumext_tmp:n {#1} }
```

(*End of definition for* `\c@__enumext_resume_i_int` *and others.*)

13.12 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.

Direct support for this is provided since LaTeX release 2025-06-01[13], but we will keep the original implementation so as not to hinder the internal *"label and ref"* system.

__enumext_register_default_label_wd:Nn

These ⟨*counters*⟩ will be used as default ⟨*labels*⟩ if the label key is not used for the different levels of the enumext, enumext*, keyans and keyans* environments, so it is necessary to get a default value for labelwidth from these ⟨*labels*⟩ at the same time.

```
\cs_new_protected:Npn \__enumext_register_default_label_wd: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_default_label_wd:Nn \arabic { 0 }
\__enumext_register_default_label_wd:Nn \Alph   { M }
\__enumext_register_default_label_wd:Nn \alph   { m }
\__enumext_register_default_label_wd:Nn \Roman  { VIII }
\__enumext_register_default_label_wd:Nn \roman  { viii }
```

(*End of definition for* __enumext_register_default_label_wd: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.)

13.13 Setting keys associated with label

When *tagged* PDF is active \makelabel is redefined using \makebox to work correctly (§13.37). From the user side it is convenient to have a key that allows using this redefinition with \makebox without having \IfDocumentMetadataTF active.

mode-box

We define the key mode-box only for the *"first level"* of enumext and enumext* environments.

```
\cs_set_protected:Npn \__enumext_tmp:n #1
  {
    \keys_define:nn { enumext /  #1 }
      {
        mode-box .bool_set:N = \l__enumext_mode_box_bool,
        mode-box .initial:n  = false,
```

```
      mode-box .value_forbidden:n = true,
    }
  }
\clist_map_inline:nn { level-1, enumext* } { \__enumext_tmp:n {#1} }
```

(*End of definition for* `mode-box`.)

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.*)

align

The `align` key is implemented differently for *"starred"* and *"non starred"* environments. For compatibility with *tagged* PDF we must set `\l__enumext_align_label_pos_X_str`.

```
\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 }
                            \str_set:cn { l__enumext_align_label_pos_#2_str } { l }
                          },
        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 }
                            \str_set:cn { l__enumext_align_label_pos_#2_str } { r }
                          },
        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 }
                            \str_set:cn { l__enumext_align_label_pos_#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
  {
    {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`.)

13.14 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.

13.14.1 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_standard_ref:n` function will first pass the key *argument* `ref` to the variable `\l__enumext_ref_key_arg_tl` and analyze its state, if it is not *empty* it will set a copy of of the *current counter style* save in `\l__enumext_the_counter_X_tl` to `\l__enumext_ref_the_count_tl` and then set the variable `\l__enumext_renew_counter_X_tl` which will modify `\theenumX`.

```
\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_the_counter_ \__enumext_level: _tl
          }
        \tl_set:ce { l__enumext_renew_counter_ \__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_counter_ \__enumext_level: _tl }
      {
        \tl_use:c { l__enumext_renew_counter_ \__enumext_level: _tl }
      }
  }
```

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

13.14.2 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_the_counter_vii_tl
            \tl_set:Ne \l__enumext_renew_counter_vii_tl
              {
                \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl } { \exp_not:V
              }
          }
      }
    \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_the_counter_viii_tl
            \tl_set:Ne \l__enumext_renew_counter_viii_tl
              {
                \exp_not:N \renewcommand  { \exp_not:V \l__enumext_ref_the_count_tl } { \exp_not:
              }
          }
```

```
    }
  }
```

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_counter_vii_tl
          {
            \tl_use:N \l__enumext_renew_counter_vii_tl
          }
      }
    \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
      {
        \tl_if_empty:NF \l__enumext_renew_counter_viii_tl
          {
            \tl_use:N \l__enumext_renew_counter_viii_tl
          }
      }
  }
```

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

13.14.3 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` if it has not been established 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}
                         \__enumext_label_style:cvn { l__enumext_label_vi_tl }
                           { l__enumext_counter_vi_tl } {#1}
                         \dim_set_eq:NN
                           \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_the_counter_v_tl
        \tl_put_right:Ne \l__enumext_renew_counter_v_tl
          {
            \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl } { \exp_not:V \l_
          }
      }
  }
```

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_counter_v_tl
      {
        \tl_use:N \l__enumext_renew_counter_v_tl
      }
```

```
}
```

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

13.15 Setting start, start* and widest keys

`\__enumext_start_from:NNn`
`\__enumext_start_from:ccn`
`\__enumext_start_from:cce`

The function `\__enumext_start_from:NNn` used by `start` and `start*` keys take three arguments:

`#1:` `\l__enumext_label_X_tl`
`#2:` `\l__enumext_start_X_int`
`#3:` ⟨*integer or string*⟩

The first argument of this function are the *"counter style"* set by `label` key, the second argument is returned by the function, the third argument can be an ⟨*integer*⟩ or ⟨*string*⟩ of the form `\Alph`, `\alph`, `\Roman` or `\roman`. This effectively allows `start*=A` or `start=1` to be used.

In version 1.6 it is allowed to pass the `resume` key *without value* by means of the command `\setenumext`, for the correct operation of this we must set the boolean variable `\l__enumext_resume_count_bool` set by the `resume` key *without value* to *"false"* (§13.26). This is necessary to be able to "reset" the *start value* by means of the `start` or `start*` keys.

```
\cs_new_protected:Npn \__enumext_start_from:NNn #1 #2 #3
  {
    \bool_set_false:N \l__enumext_resume_count_bool
    \__enumext_if_is_int:nTF { #3 }
       {
         \int_set:Nn #2 {#3}
       }
       {
         \regex_if_match:nVT { \c{Alph} | \c{alph} } {#1}
           { \int_set:Nn #2 { \int_from_alph:n {#3} } }
         \regex_if_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, cce }
```

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

`\__enumext_widest_from:nNNn`
`\__enumext_widest_from:nccn`

The function `\__enumext_widest_from:nNNn` used by the `widest` key take four arguments:

`#1:` The counter associated with the environment level
`#2:` `\l__enumext_label_X_tl`
`#3:` `\l__enumext_labelwidth_X_dim`
`#4:` ⟨*integer or string*⟩

The second and third arguments of this function are the values set by `label` and `labelwidth` keys, the four argument can be an ⟨*integer*⟩ or ⟨*string*⟩ of the form `\Alph`, `\alph`, `\Roman` or `\roman`. The value of the four argument is set temporarily for the identified counter in this point (level), then the value is expanded into a *"box"* and the *"width"* of the *"box"* is returned.

```
\cs_new_protected:Npn \__enumext_widest_from:nNNn #1 #2 #3 #4
  {
    \__enumext_if_is_int:nTF {#4}
      {
        \setcounter{enumX#1} { #4 }
      }
      {
        \regex_if_match:nVT { \c{Alph} | \c{alph} } {#2}
          { \setcounter{enumX#1} { \int_from_alph:n {#4} } }
        \regex_if_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`
`start*`
`widest`

Now define and set `start*`, `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* .value_required:n = true,
        start  .code:n    = {
                                \__enumext_start_from:cce
                                  { l__enumext_label_#2_tl }
                                  { l__enumext_start_#2_int } { \int_eval:n {##1} }
                              },
        start  .initial:n = 1,
        start  .value_required:n = true,
        widest .code:n    = {
                                \__enumext_widest_from:nccn {#2}
                                  { l__enumext_label_#2_tl }
                                  { l__enumext_labelwidth_#2_dim } {##1}
                              },
        widest .value_required:n = true,
      }
  }
\clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }
```

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

13.16 Setting keys for penaltys

beginpenalty
midpenalty
endpenalty

The three parameters \@beginparpenalty, \@itempenalty and \@endparpenalty work together to ensure that list environments look good, avoiding unsightly page breaks that can break the flow of the list, so it's a good idea to have a ⟨*keys*⟩ to access these.

```
\cs_set_protected:Npn \__enumext_tmp:nn #1 #2
  {
    \keys_define:nn { enumext / #1 }
      {
        beginpenalty .int_set:c = { l__enumext_beginparpenalty_#2_int },
        beginpenalty .initial:n = -51,
        beginpenalty .value_required:n = true,
        midpenalty   .int_set:c = { l__enumext_itempenalty_#2_int },
        midpenalty   .initial:n = -51,
        midpenalty   .value_required:n = true,
        endpenalty   .int_set:c = { l__enumext_endparpenalty_#2_int },
        endpenalty   .initial:n = -51,
        endpenalty   .value_required:n = true,
      }
  }
\clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }
```

(*End of definition for* `beginpenalty`, `midpenalty`, *and* `endpenalty`.)

13.17 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.*)

13.18 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:` apply `\vspace{-\baselineskip}` and set `\topsep=0pt` for the *"first level"* of the nested `enumext` environment.

`base-fix`
`\__enumext_nested_base_line_fix:`

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

```
\keys_define:nn { enumext /  level-1 }
  {
    base-fix .bool_set:N = \l__enumext_base_line_fix_bool,
    base-fix .initial:n  = false,
    base-fix .value_forbidden:n = true,
  }
```

The function `\__enumext_nested_base_line_fix:` passed to the `\__enumext_parse_keys:n` function in the definition of the `enumext` environment (§13.42) will be responsible for applying the *baseline correction* and adjusting the ⟨*keys*⟩ for the `enumext` environment and the `\printkeyans` with *starred argument* '`*`' (§13.50).

We will first implement the function code from the user side of the `base-fix` key, that is, only the user knows when it is necessary to apply it within the document in which case the variable `\l__enumext_print_keyans_star_bool` set by the `\printkeyans` command is false and the variable `\l__enumext_base_line_fix_bool` is true.

We set the values of the keys `topsep`, `above` and `above*` for the *"first level"* of `enumext` environment equal to `0pt` and finally set the variable `\l__enumext_base_line_fix_bool` to false.

```
\cs_new_protected:Nn \__enumext_nested_base_line_fix:
  {
    \bool_lazy_all:nT
      {
        { \bool_if_p:N \l__enumext_starred_first_bool }
        { \bool_if_p:N \l__enumext_base_line_fix_bool }
        { \bool_not_p:n { \l__enumext_print_keyans_star_bool } }
      }
      {
        \mode_leave_vertical:
        \vspace { -\dim_eval:n { \baselineskip + \parsep } }
        \keys_set:nn { enumext / level-1 }
          {
            topsep = 0pt, above = 0pt, above* = 0pt,
          }
      }
```

When we are running the \printkeyans command with the *starred argument* '*' the variable \l__enumext_print_keyans_star_bool is true and we can run a simplified version of \vspace using \skip_vertical:n.

```
    \bool_lazy_and:nnT
      { \bool_if_p:N \l__enumext_starred_first_bool }
      { \bool_if_p:N \l__enumext_print_keyans_star_bool }
      {
        \mode_leave_vertical:
        \skip_vertical:n { -\baselineskip }
        \skip_vertical:N \c_zero_skip
        \keys_set:nn { enumext / level-1 }
          {
            topsep = 0pt, above = 0pt, above* = 0pt,
          }
      }
    \bool_set_false:N \l__enumext_base_line_fix_bool
  }
```

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

13.19 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 }
```

13.19.1 Functions for setting the fake itemindent

__enumext_fake_item_indent:
__enumext_keyans_fake_item_indent:
__enumext_fake_item_indent_vii:
__enumext_fake_item_indent_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_indent:
  {
    \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 } }
            \exp_not:N \ignorespaces
          }
      }
  }
\cs_set_protected:Nn \__enumext_keyans_fake_item_indent:
  {
    \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
            \exp_not:N \ignorespaces
          }
      }
  }
\cs_set_protected:Nn \__enumext_fake_item_indent_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 \skip_horizontal:N \l__enumext_fake_item_indent_vii_dim
            \exp_not:N \ignorespaces
          }
      }
  }
\cs_set_protected:Nn \__enumext_fake_item_indent_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 \skip_horizontal:N \l__enumext_fake_item_indent_viii_dim
            \exp_not:N \ignorespaces
          }
      }
  }
```

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

13.20 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.)

13.21 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.*)

13.21.1 Functions for before, after and first keys in enumext

__enumext_before_args_exec:
__enumext_before_keys_exec:
__enumext_after_stop_list:
__enumext_after_args_exec:

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

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

```
\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: \endlist{⟨*code*⟩}.

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

13.21.2 Functions for before, after and first keys in keyans

__enumext_before_args_exec_v:
__enumext_before_keys_exec_v:
__enumext_after_stop_list_v:
__enumext_after_args_exec_v:

Same implementation as the one used in the enumext environment.

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

```
    \tl_use:N \l__enumext_before_no_starred_key_v_tl
  }
\cs_new_protected:Nn \__enumext_after_stop_list_v:
  {
    \tl_use:N \l__enumext_after_stop_list_v_tl
  }
\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.*)

13.21.3 Functions for `before`, `after` and `first` keys in `enumext*` and `keyans*`

`\__enumext_before_args_exec_vii:`
`\__enumext_before_keys_exec_vii`
`\__enumext_after_stop_list_vii:`
`\__enumext_after_args_exec_vii:`

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

```
\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
  }
\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
  }
\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
  }
\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.*)

13.22 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.*)

13.23 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.

13.23.1 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:`.)

13.23.2 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:`.)

13.24 Adjustment of vertical spaces for `minipage`

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

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

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

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

Again I must make clear that the implementation here is a *"bit questionable"*, but hunting the spaces (`glue`) produced by the `minipage` environment is quite complicated, even more if `multicols` it is nested. The setting of the values was more *"trial and error"* (approx to `\strutbox`), using the help of the `lua-visual-debug`[15] package, again my attempts to find the correct values using `\showoutput` and `\showboxdepth` absolutely failed.

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

`\__enumext_minipage_set_skip:`
`\__enumext_minipage_add_space:`

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

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

```
\cs_new_protected:Nn \__enumext_minipage_set_skip:
  {
    \skip_set:Nn \l__enumext_minipage_right_skip
      {
        \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
      }
    \mode_if_vertical:T
      {
        \skip_add:Nn \l__enumext_minipage_right_skip
          {
            \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
          }
      }
    \skip_set_eq:NN \l__enumext_minipage_after_skip \l__enumext_minipage_right_skip
```

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

```
    \skip_set_eq:cN
      { l__enumext_multicols_above_ \__enumext_level: _skip } \l__enumext_minipage_right_skip
    \skip_set_eq:cN
      { l__enumext_multicols_below_ \__enumext_level: _skip } \l__enumext_minipage_right_skip
    \__enumext_pre_itemsep_skip:
```

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

```
    \int_compare:nNnT
      { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } > { 1 }
      {
        \skip_zero:N \topskip
        \skip_set_eq:Nc \multicolsep { l__enumext_multicols_above_ \__enumext_level: _skip }
      }
  }
```

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

```
\cs_new_protected:Nn \__enumext_minipage_add_space:
  {
    \__enumext_minipage_set_skip:
    \__enumext_unskip_unkern:
    \mode_if_vertical:TF
      {
        \nopagebreak\nointerlineskip
      }
      {
        \par\nopagebreak\nointerlineskip
        \skip_zero:c { l__enumext_partopsep_ \__enumext_level: _skip }
      }
    \int_compare:nNnTF
      { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } > { 1 }
      {
        \addvspace{ 0.445\box_ht:N \strutbox }
      }
      {
        \addvspace{ 0.250\box_ht:N \strutbox }
      }
  }
```

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

__enumext_pre_itemsep_skip:

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

```
\cs_new_protected:Nn \__enumext_pre_itemsep_skip:
  {
    \int_case:nn { \l__enumext_level_int }
      {
        { 2 }{
               \skip_if_eq:nnTF
                 { \l__enumext_itemsep_i_skip } { \l__enumext_minipage_after_skip }
                 {
                   \skip_set:Nn \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
                   \skip_set:Nn \l__enumext_multicols_below_ii_skip { 0.350\box_ht:N \strutbox }
                 }
                 {
                   \dim_compare:nNnT
                     { \l__enumext_itemsep_i_skip } < { \l__enumext_minipage_after_skip }
                     {
                       \skip_sub:Nn
                         \l__enumext_minipage_after_skip { \l__enumext_itemsep_i_skip }
                       \skip_sub:Nn
                         \l__enumext_multicols_below_ii_skip { \l__enumext_itemsep_i_skip }
                       \skip_add:Nn
                         \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
```

```
                \skip_add:Nn
                  \l__enumext_multicols_below_ii_skip { 0.350\box_ht:N \strutbox }
              }
            \dim_compare:nNnT
              { \l__enumext_itemsep_i_skip } > { \l__enumext_minipage_after_skip }
              {
                \skip_set:Nn \l__enumext_minipage_temp_skip
                  {
                    \l__enumext_itemsep_i_skip - \l__enumext_minipage_after_skip
                  }
                \skip_sub:Nn
                  \l__enumext_minipage_after_skip { \l__enumext_itemsep_i_skip }
                \skip_sub:Nn
                  \l__enumext_multicols_below_ii_skip { \l__enumext_itemsep_i_skip }
                \skip_add:Nn
                  \l__enumext_minipage_after_skip
                  { 0.150\box_ht:N \strutbox + \l__enumext_minipage_temp_skip }
                \skip_add:Nn
                  \l__enumext_multicols_below_ii_skip
                  { 0.350\box_ht:N \strutbox + \l__enumext_minipage_temp_skip }
              }
          }
      }
    { 3 }{
        \skip_if_eq:nnTF
          { \l__enumext_itemsep_ii_skip } { \c_zero_skip }
          {
            \skip_set:Nn \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
            \skip_set:Nn \l__enumext_multicols_below_iii_skip { 0.350\box_ht:N \strutbox }
          }
          {
            \dim_compare:nNnT
              { \l__enumext_itemsep_ii_skip } < { \l__enumext_minipage_after_skip }
              {
                \skip_sub:Nn
                  \l__enumext_minipage_after_skip { \l__enumext_itemsep_ii_skip }
                \skip_sub:Nn
                  \l__enumext_multicols_below_iii_skip { \l__enumext_itemsep_ii_skip }
                \skip_add:Nn
                  \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
                \skip_add:Nn
                  \l__enumext_multicols_below_iii_skip { 0.350\box_ht:N \strutbox }
              }
            \dim_compare:nNnT
              { \l__enumext_itemsep_ii_skip } > { \l__enumext_minipage_after_skip }
              {
                \skip_set:Nn \l__enumext_minipage_temp_skip
                  {
                    \l__enumext_itemsep_ii_skip - \l__enumext_minipage_after_skip
                  }
                \skip_sub:Nn
                  \l__enumext_minipage_after_skip { \l__enumext_itemsep_ii_skip }
                \skip_sub:Nn
                  \l__enumext_multicols_below_iii_skip { \l__enumext_itemsep_ii_skip }
                \skip_add:Nn
                  \l__enumext_minipage_after_skip
                  { 0.150\box_ht:N \strutbox + \l__enumext_minipage_temp_skip }
                \skip_add:Nn
                  \l__enumext_multicols_below_iii_skip
                  { 0.350\box_ht:N \strutbox + \l__enumext_minipage_temp_skip }
              }
          }
      }
    { 4 }{
        \skip_if_eq:nnTF { \l__enumext_itemsep_iii_skip } { \c_zero_skip }
          {
            \skip_set:Nn \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
            \skip_set:Nn \l__enumext_multicols_below_iv_skip { 0.350\box_ht:N \strutbox }
          }
          {
            \dim_compare:nNnT
```

```
                    { \l__enumext_itemsep_iii_skip } < { \l__enumext_minipage_after_skip }
                    {
                      \skip_sub:Nn
                        \l__enumext_minipage_after_skip { \l__enumext_itemsep_iii_skip }
                      \skip_sub:Nn
                        \l__enumext_multicols_below_iv_skip { \l__enumext_itemsep_iii_skip }
                      \skip_add:Nn
                        \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
                      \skip_add:Nn
                        \l__enumext_multicols_below_iv_skip { 0.350\box_ht:N \strutbox }
                    }
                  \dim_compare:nNnT
                    { \l__enumext_itemsep_iii_skip } > { \l__enumext_minipage_after_skip }
                    {
                      \skip_set:Nn \l__enumext_minipage_temp_skip
                        {
                          \l__enumext_itemsep_iii_skip - \l__enumext_minipage_after_skip
                        }
                      \skip_sub:Nn
                        \l__enumext_minipage_after_skip { \l__enumext_itemsep_iii_skip }
                      \skip_sub:Nn
                        \l__enumext_multicols_below_iv_skip { \l__enumext_itemsep_iii_skip }
                      \skip_add:Nn
                        \l__enumext_minipage_after_skip
                        { 0.150\box_ht:N \strutbox + \l__enumext_minipage_temp_skip }
                      \skip_add:Nn
                        \l__enumext_multicols_below_iv_skip
                        { 0.350\box_ht:N \strutbox + \l__enumext_minipage_temp_skip }
                    }
                }
            }
        }
    }
```

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

13.24.2 Adjustment of vertical spaces for `minipage` in keyans

`\__enumext_keyans_minipage_set_skip:`
`\__enumext_keyans_minipage_add_space:`
`\__enumext_keyans_pre_itemsep_skip:`

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

```
\cs_new_protected:Nn \__enumext_keyans_minipage_set_skip:
  {
    \skip_zero:N \l__enumext_minipage_after_skip
    \skip_zero:N \l__enumext_minipage_left_skip
    \skip_zero:N \l__enumext_minipage_right_skip
    \skip_set:Nn \l__enumext_minipage_right_skip
      {
        \l__enumext_topsep_v_skip
      }
    \mode_if_vertical:T
      {
        \skip_add:Nn \l__enumext_minipage_right_skip
          {
            \l__enumext_partopsep_v_skip
          }
      }
    \skip_set_eq:NN \l__enumext_minipage_after_skip \l__enumext_minipage_right_skip
    \skip_set_eq:NN \l__enumext_multicols_above_v_skip \l__enumext_minipage_right_skip
    \skip_set_eq:NN \l__enumext_multicols_below_v_skip \l__enumext_minipage_right_skip
    \__enumext_keyans_pre_itemsep_skip:
    \int_compare:nNnT { \l__enumext_columns_v_int } > { 1 }
      {
        \skip_zero:N \topskip
        \skip_set_eq:NN \multicolsep \l__enumext_minipage_right_skip
      }
  }
\cs_new_protected:Nn \__enumext_keyans_minipage_add_space:
  {
    \__enumext_keyans_minipage_set_skip:
    \__enumext_unskip_unkern:
    \mode_if_vertical:TF
```

```
      {
        \nopagebreak\nointerlineskip
      }
      {
        \par\nopagebreak\nointerlineskip
        \skip_zero:N \l__enumext_partopsep_v_skip
      }
    \int_compare:nNnTF { \l__enumext_columns_v_int } > { 1 }
      {
        \addvspace{ 0.445\box_ht:N \strutbox }
      }
      {
        \addvspace{ 0.250\box_ht:N \strutbox }
      }
  }
\cs_new_protected:Nn \__enumext_keyans_pre_itemsep_skip:
  {
    \skip_if_eq:nnTF
      { \l__enumext_itemsep_i_skip } { \l__enumext_minipage_after_skip }
      {
        \skip_set:Nn \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
        \skip_set:Nn \l__enumext_multicols_below_v_skip { 0.350\box_ht:N \strutbox }
      }
      {
        \dim_compare:nNnT
          { \l__enumext_itemsep_i_skip } < { \l__enumext_minipage_after_skip }
          {
            \skip_sub:Nn \l__enumext_minipage_after_skip { \l__enumext_itemsep_i_skip }
            \skip_sub:Nn \l__enumext_multicols_below_v_skip { \l__enumext_itemsep_i_skip }
            \skip_add:Nn \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
            \skip_add:Nn \l__enumext_multicols_below_v_skip { 0.350\box_ht:N \strutbox }
          }
        \dim_compare:nNnT
          { \l__enumext_itemsep_i_skip } > { \l__enumext_minipage_after_skip }
          {
            \skip_set:Nn \l__enumext_minipage_temp_skip
              {
                \l__enumext_itemsep_i_skip - \l__enumext_minipage_after_skip
              }
            \skip_sub:Nn \l__enumext_minipage_after_skip { \l__enumext_itemsep_i_skip }
            \skip_sub:Nn \l__enumext_multicols_below_v_skip { \l__enumext_itemsep_i_skip }
            \skip_add:Nn \l__enumext_minipage_after_skip
              { 0.150\box_ht:N \strutbox + \l__enumext_minipage_temp_skip }
            \skip_add:Nn \l__enumext_multicols_below_v_skip
              { 0.350\box_ht:N \strutbox + \l__enumext_minipage_temp_skip }
          }
      }
  }
```

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

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

`\__enumext_mini_set_vskip_vii:`
`\__enumext_mini_set_vskip_viii:`

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

```
\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_page` environment on the *left side* when the `mini-right` key is active in the `enumext*` and `keyans*` environments.
Here we will NOT take into account whether TEX is in ⟨*horizontal mode*⟩ or ⟨*vertical mode*⟩, since `\partopsep` is equal to `0pt` in both environments.

```
\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:`.)

13.24.4 The command `\miniright`

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

\miniright

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

```
\NewDocumentCommand \miniright { s }
  {
    \int_compare:nNnT { \l__enumext_keyans_pic_level_int } = { 1 }
      {
        \msg_error:nnn { enumext } { wrong-miniright-place }
      }
    % outside
    \bool_lazy_and:nnT
      { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
      { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
      {
        \msg_error:nnn { enumext } { wrong-miniright-place }
      }
    % starred env
    \bool_lazy_and:nnT
      { \bool_if_p:N \g__enumext_starred_bool }
      { \bool_not_p:n { \l__enumext_standar_bool } }
      {
        \msg_error:nnn { enumext } { wrong-miniright-starred }
      }
    % exec
    \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* 12.)

__enumext_mini_right_cmd:n

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

```
\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:
        \int_compare:nNnT
          { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } = { 1 }
          {
            \par\addvspace{ \l__enumext_minipage_after_skip }
          }
        \end__enumext_mini_page
        \hfill
        \__enumext_mini_page{ \dim_use:c { l__enumext_minipage_right_ \__enumext_level: _dim } }
          \par\nointerlineskip
          \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 } }
    % paranoia
    \RenewDocumentCommand \miniright { s }
      {
        \msg_error:nn { enumext } { many-miniright-used }
      }
  }
```

(*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:
        \int_compare:nNnT { \l__enumext_columns_v_int } = { 1 }
          {
            \par\addvspace{ \l__enumext_minipage_after_skip }
          }
        \end__enumext_mini_page
        \hfill
        \__enumext_mini_page{ \l__enumext_minipage_right_v_dim }
          \par\nointerlineskip
          \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 } }
    % paranoia
    \RenewDocumentCommand \miniright { s }
      {
        \msg_error:nn { enumext } { many-miniright-used }
      }
  }
```

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

13.25 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.*)

13.25.1 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:`.)

13.25.2 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:`.)

13.25.3 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:`.)

13.26 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 environments `enumext` and `enumext*`, but, discarding some specific ⟨*keys*⟩.

`series` `resume` `resume*`

We define the keys `series`, `resume` and `resume*` for the *"all levels"* of `enumext` and `enumext*`. Here we do not need to make sure that `\printkeyans` is not running otherwise the *start value* of the environments would be increased when using `resume` or `resume*` keys.
In version 1.6 it is allowed to pass the key `resume` *without value* by means of the command `\setenumext`, for the correct operation of this we must set the boolean variable `\l__enumext_resume_count_bool` set by the key `resume` *without value* to "true" to be later processed by the function `\__enumext_parse_series:n` in the definition of the environments `enumext` and `enumext*`.

```
\cs_set_protected:Npn \__enumext_tmp:nn #1 #2
  {
    \keys_define:nn { enumext / #1 }
      {
        series  .str_set:N = \l__enumext_series_name_str,
        series  .value_required:n = true,
        resume  .code:n = {
                            \bool_if:NF \l__enumext_print_keyans_cmd_bool
                              {
                                \tl_set:Nn \l__enumext_series_name_tl {##1}
                                \tl_if_empty:NTF \l__enumext_series_name_tl
```

```
                                    {
                                      \bool_set_true:c { l__enumext_resume_count_#2_bool }
                                      \bool_set_eq:Nc
                                        \l__enumext_resume_count_bool
                                        { l__enumext_resume_count_#2_bool }
                                    }
                                    {
                                      \__enumext_resume:n {##1}
                                    }
                                }
                            },
        resume* .code:n = {
                                \bool_if:NF \l__enumext_print_keyans_cmd_bool
                                  {
                                    \bool_set_true:c { l__enumext_resume_count_#2_bool }
                                    \bool_set_eq:Nc
                                      \l__enumext_resume_count_bool { l__enumext_resume_count_#2_bool
                                    \bool_set_true:c { l__enumext_resume_star_key_#2_bool }
                                    \__enumext_resume_star:
                                  }
                             },
        resume* .value_forbidden: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* `series`, `resume`, *and* `resume*`.)

13.26.1 Internal function to save counter and integer values

`\__enumext_standar_save_counter:`
`\__enumext_standar_save_counter_aux:`
`\__enumext_starred_save_counter:`
`\__enumext_starred_save_counter_aux:`

The `\__enumext_standar_save_counter:` and `\__enumext_starred_save_counter:` functions will save the last counter value to `\g__enumext_series_`⟨*series name*⟩`_int` if the `series`={⟨*series name*⟩} key has been passed, to `\c@__enumext_resume_X_int` if it has passed the key `resume` *without value* and the key `series` is not active, in `\g__enumext_series_`⟨*series name*⟩`_X_int` if the key `resume`={⟨*series name*⟩} has been passed and in `\g__enumext_series_`⟨*store name*⟩`_X_int` if the key has been passed `save-ans`={⟨*store name*⟩}.

The variables `\l__enumext_series_name_str` and `\l__enumext_series_name_tl` contain the same {⟨*series name*⟩} but are executed at different moments, the integer variable with `\l__enumext_series_name_str` sets the value when execute `series`={⟨*series name*⟩} and the integer variable with `\l__enumext_series_name_tl` sets the subsequent values when use `resume`={⟨*series name*⟩}. This function is passed to the `enumext` environment definition (§13.42) and the `enumext*` environment definition (§13.47).

```
\cs_new_protected:Nn \__enumext_standar_save_counter:
  {
    \bool_if:NTF \g__enumext_standar_bool
      {
        \__enumext_standar_save_counter_aux:
        \int_compare:nNnT { \l__enumext_level_int } = { 1 }
          {
            \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}
              }
          }
      }
      {
        \__enumext_standar_save_counter_aux:
      }
  }
\cs_new_protected:Nn \__enumext_standar_save_counter_aux:
  {
    \str_if_empty:NF \l__enumext_series_name_str
      {
        \int_gset_eq:cc
          { g__enumext_series_ \l__enumext_series_name_str _ \__enumext_level: _int }
          { c@enumX \__enumext_level: }
      }
```

```
    \tl_if_empty:NTF \l__enumext_series_name_tl
      {
        \str_if_empty:NT \l__enumext_series_name_str
          {
            \tl_if_empty:NT \l__enumext_store_name_tl
              {
                \int_gset_eq:cc
                  { c@ __enumext_resume_ \__enumext_level: _int } { c@enumX \__enumext_level: }
              }
          }
      }
      {
        \int_if_exist:cT
          { g__enumext_series_ \l__enumext_series_name_tl _ \__enumext_level: _int }
          {
            \int_gset_eq:cc
              { g__enumext_series_ \l__enumext_series_name_tl _ \__enumext_level: _int }
              { c@enumX \__enumext_level: }
          }
      }
  }
\cs_new_protected:Nn \__enumext_starred_save_counter:
  {
    \bool_if:NTF \g__enumext_starred_bool
      {
        \__enumext_starred_save_counter_aux:
        \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}
          }
      }
      {
        \__enumext_starred_save_counter_aux:
      }
  }
\cs_new_protected:Nn \__enumext_starred_save_counter_aux:
  {
    \str_if_empty:NF \l__enumext_series_name_str
      {
        \int_gset_eq:cN
          { g__enumext_series_ \l__enumext_series_name_str _vii_int } \value{enumXvii}
      }
    \tl_if_empty:NTF \l__enumext_series_name_tl
      {
        \str_if_empty:NT \l__enumext_series_name_str
          {
            \tl_if_empty:NT \l__enumext_store_name_tl
              {
                \int_gset_eq:cc { c@ __enumext_resume_vii_int } { c@enumXvii }
              }
          }
      }
      {
        \int_if_exist:cT { g__enumext_series_ \l__enumext_series_name_tl _vii_int }
          {
            \int_gset_eq:cN
              { g__enumext_series_ \l__enumext_series_name_tl _vii_int } \value{enumXvii}
          }
      }
  }
```

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

13.26.2 Internal function for resume counters

`\__enumext_resume_counter:`

The `\__enumext_resume_counter:` function is executed by the `resume*` key and `resume` key *without value*, only the *"counters"* for the *"levels"* of the environments in which it is executed will be set. If the `save-ans` key is active it will set the *"counter"* according to the value of the integer variable created by that key.

```
\cs_new_protected:Nn \__enumext_resume_counter:
  {
    \cs_set:Npn \__enumext_tmp:n ##1
```

```
      {
        \bool_if:cT { l__enumext_resume_count_ \int_to_roman:n {##1} _bool }
          {
            \exp_args:Ne \int_set:cn { l__enumext_start_ \int_to_roman:n {##1} _int }
              {
                \int_use:c { c@ __enumext_resume_ \int_to_roman:n {##1} _int } + 1
              }
          }
      }
    \int_compare:nNnT { \l__enumext_level_int } > { 0 }
      {
        \bool_lazy_and:nnTF
           { \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
               }
           }
           {
             \int_step_function:nN { \l__enumext_level_int } \__enumext_tmp:n
           }
      }
    \int_compare:nNnT { \l__enumext_level_h_int } = { 1 }
      {
        \bool_lazy_and:nnTF
          { \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
              }
          }
          {
            \bool_if:NT \l__enumext_resume_count_vii_bool
              {
                \int_set:Nn \l__enumext_start_vii_int
                  {
                    \int_use:c { c@ __enumext_resume_vii_int } + 1
                  }
              }
          }
      }
  }
```

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

13.26.3 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 argument* passed to the environment. This implementation is adapted directly from the code provided by Jonathan P. Spratte (@Skillmon) in chat-TeX-SX

```
\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*`, `reset`, `reset*` and `base-fix` keys.

```
\cs_new:Npn \__enumext_filter_series_key:n #1
  {
    \str_case:nnF {#1}
      {
        { resume } {} { resume* } {} { reset } {} { reset* } {} { 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, start*, save-ans and save-key keys.

```
\cs_new:Npn \__enumext_filter_series_pair:nn #1#2
  {
    \str_case:nnF {#1}
      {
        { series } {} { resume } {} { start } {}
        { 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_save_last_keys:n
__enumext_resume_last_counter:

The function __enumext_save_last_keys:n will be in charge of saving the filtering ⟨*keys*⟩ when the keys series={⟨*series name*⟩} or resume={⟨*series name*⟩} or resume* are NOT active and will save them in the variable \g__enumext_resume_last_keys_X_tl for the enumext environment and in the variable \g__enumext_resume_last_keys_vii_tl for the enumext* environment.

The boolean variable \l__enumext_resume_series_X_bool is set to "true" by the key resume={⟨*series name*⟩}, the boolean variable \l__enumext_resume_star_key_X_bool is set to "true" by the key resume*, in this case we need to make sure both variables are set to ""false" so that they don't override the default filtered ⟨*keys*⟩.

```
\cs_new_protected:Npn \__enumext_save_last_keys:n #1
  {
    \int_compare:nNnT { \l__enumext_level_int } > { 0 }
      {
        \bool_if:cF { l__enumext_resume_series_ \__enumext_level: _bool }
          {
            \bool_if:cF { l__enumext_resume_star_key_ \__enumext_level: _bool }
              {
                \tl_gclear:c { g__enumext_resume_last_keys_ \__enumext_level: _tl }
                \tl_gset:ce
                  { g__enumext_resume_last_keys_ \__enumext_level: _tl }
                  { \__enumext_filter_series:n {#1} }
              }
          }
      }
    \int_compare:nNnT { \l__enumext_level_h_int } = { 1 }
      {
        \bool_if:NF \l__enumext_resume_series_vii_bool
          {
            \tl_gclear:N \g__enumext_resume_last_keys_vii_tl
            \tl_gset:Ne \g__enumext_resume_last_keys_vii_tl { \__enumext_filter_series:n {#1} }
          }
      }
  }
```

The __enumext_resume_last_counter: function will be in charge of setting the *"counters"* when the keys series={⟨*series name*⟩} or resume={⟨*series name*⟩} are NOT active and the resume key is being used *without value* either in the *optional argument* of the environments or through the \setenumext command.

The boolean variable \l__enumext_resume_count_bool is set to "true" by the keys resume *without value* and resume*; and set to "false" by the keys start and start* (§13.15).

```
\cs_new_protected:Nn \__enumext_resume_last_counter:
  {
    \bool_lazy_and:nnT
      { \bool_if_p:N \l__enumext_resume_count_bool }
      { \tl_if_empty_p:N \l__enumext_series_name_tl }
      {
        \__enumext_resume_counter:
      }
  }
```

(*End of definition for* __enumext_save_last_keys:n *and* __enumext_resume_last_counter:.)

__enumext_parse_series:n

The __enumext_parse_series:n function handled by the series key will be responsible for *storing* the *filtered* ⟨*keys*⟩ from the *optional arguments* of the enumext and enumext* environments for the resume and resume* keys. If the series key is NOT active it will call the __enumext_save_last_keys:n function

to *store* the *filtered* ⟨*keys*⟩ that will be used by the `resume*` key and then the `\__enumext_resume_last_counter:` function used by the `resume` key *without value* if it is active, otherwise *store* the *filtered* ⟨*keys*⟩ in the global variable `\g__enumext_series_`⟨*series name*⟩`_X_tl` along with the creation of the integer variable `\g__enumext_series_`⟨*series name*⟩`_X_int` used by the `resume` key *with value*.

This function is passed to the function `\__enumext_parse_keys:n` in the `enumext` environment definition (§13.42) and to the function `\__enumext_parse_keys_vii:n` in the `enumext*` environment definition (§13.47).

```
\cs_new_protected:Npn \__enumext_parse_series:n #1
  {
    \str_if_empty:NTF \l__enumext_series_name_str
      {
        \int_compare:nNnT { \l__enumext_level_int } > { 0 }
          {
            \__enumext_save_last_keys:n {#1}
            \__enumext_resume_last_counter:
          }
        \int_compare:nNnT { \l__enumext_level_h_int } = { 1 }
          {
            \__enumext_save_last_keys:n {#1}
            \__enumext_resume_last_counter:
          }
      }
      {
        \int_compare:nNnT { \l__enumext_level_int } > { 0 }
          {
            \tl_gclear_new:c
              { g__enumext_series_ \l__enumext_series_name_str _ \__enumext_level: _tl }
            \tl_gset:ce
              { g__enumext_series_ \l__enumext_series_name_str _ \__enumext_level: _tl }
              { \__enumext_filter_series:n {#1} }
            \int_if_exist:cF
              { g__enumext_series_ \l__enumext_series_name_str _ \__enumext_level: _int }
              {
                \int_new:c
                  { g__enumext_series_ \l__enumext_series_name_str _ \__enumext_level: _int }
              }
          }
        \int_compare:nNnT { \l__enumext_level_h_int } = { 1 }
          {
            \tl_gclear_new:c { g__enumext_series_ \l__enumext_series_name_str _vii_tl }
            \tl_gset:ce
              { g__enumext_series_ \l__enumext_series_name_str _vii_tl }
              { \__enumext_filter_series:n {#1} }
            \int_if_exist:cF { g__enumext_series_ \l__enumext_series_name_str _vii_int }
              {
                \int_new:c { g__enumext_series_ \l__enumext_series_name_str _vii_int }
              }
          }
      }
  }
```

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

13.26.4 Internal functions for `resume` key with value

`\__enumext_resume:n`

The function `\__enumext_resume:n` will handle the *argument* {⟨*series name*⟩} passed to the `resume` key in `enumext` and `enumext*` environments. First we will check if the global variable `\g__enumext_series_`⟨*series name*⟩`_X_tl` exists, if so we will call the function `\__enumext_resume_series:n` and pass the ⟨*keys*⟩ stored in `\g__enumext_series_`⟨*series name*⟩`_X_tl` to the environments, otherwise we will return an error.

```
\cs_new_protected:Npn \__enumext_resume:n #1
  {
    \int_compare:nNnT { \l__enumext_level_int } > { 0 }
      {
        \tl_if_exist:cTF { g__enumext_series_ \tl_to_str:n {#1} _ \__enumext_level: _tl }
          {
            \__enumext_resume_series:n {#1}
            \exp_args:Ne \keys_set:nv { enumext / level-\int_use:N \l__enumext_level_int }
              { g__enumext_series_ \tl_to_str:n {#1} _ \__enumext_level: _tl }
          }
          {
            \msg_error:nnn { enumext } { unknown-series-standar } {#1}
          }
```

```
      }
    \int_compare:nNnT { \l__enumext_level_h_int } = { 1 }
      {
        \tl_if_exist:cTF { g__enumext_series_ \tl_to_str:n {#1} _vii_tl }
          {
            \__enumext_resume_series:n {#1}
            \keys_set:nv { enumext / enumext* }
              { g__enumext_series_ \tl_to_str:n {#1} _vii_tl }
          }
          {
            \msg_error:nnn { enumext } { unknown-series-starred } {#1}
          }
      }
  }
```

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

`\__enumext_resume_series:n`
`\__enumext_resume_integer_series:`

The function `\__enumext_resume_series:n` will set the variable `\l__enumext_resume_series_X_bool` to "true" and pass the {⟨*argument*⟩} to the variable `\l__enumext_series_name_tl` then call the function `\__enumext_resume_integer_series:`.

```
\cs_new_protected:Npn \__enumext_resume_series:n #1
  {
    \int_compare:nNnT { \l__enumext_level_int } > { 0 }
      {
        \bool_set_true:c { l__enumext_resume_series_ \__enumext_level: _bool }
        \tl_clear:N \l__enumext_series_name_tl
        \tl_set:Nn \l__enumext_series_name_tl {#1}
        \__enumext_resume_integer_series:
      }
    \int_compare:nNnT { \l__enumext_level_h_int } = { 1 }
      {
        \bool_set_true:N \l__enumext_resume_series_vii_bool
        \tl_clear:N \l__enumext_series_name_tl
        \tl_set:Nn \l__enumext_series_name_tl {#1}
        \__enumext_resume_integer_series:
      }
  }
```

The function `\__enumext_resume_integer_series:` will be executed when the `resume`={⟨*series name*⟩} key is active, setting the *start value* for the *"counter"* of the *"current level"* of the environments in which it is run according to the value of the *"integer variables"* created by the `series` key. If the `save-ans` key is active it will set the *start value* for the *"counter"* according to the value of the integer variable created by that key.

```
\cs_new_protected:Nn \__enumext_resume_integer_series:
  {
    \cs_set:Npn \__enumext_tmp:n ##1
      {
        \int_if_exist:cT { g__enumext_series_ \l__enumext_series_name_tl _ \int_to_roman:n {##1} _
          {
            \exp_args:Ne \int_set:cn { l__enumext_start_ \int_to_roman:n {##1} _int }
              {
                \int_use:c { g__enumext_series_ \l__enumext_series_name_tl _ \int_to_roman:n {##1
              }
          }
      }
    \int_compare:nNnT { \l__enumext_level_int } > { 0 }
      {
        \bool_lazy_and:nnTF
          { \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
              }
          }
          {
            \int_step_function:nN { \l__enumext_level_int } \__enumext_tmp:n
          }
      }
    \int_compare:nNnT { \l__enumext_level_h_int } = { 1 }
      {
```

```
      \bool_lazy_and:nnTF
        { \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
            }
        }
        {
          \int_set:Nn \l__enumext_start_vii_int
            {
              \int_use:c { g__enumext_series_ \l__enumext_series_name_tl _vii_int } + 1
            }
        }
    }
}
```

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

13.26.5 Internal function for `resume*` key

`\__enumext_resume_star:`

The function `\__enumext_resume_star:` 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 and ⟨*keys*⟩ 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_star:
  {
    \cs_set:Npn \__enumext_tmp:n ##1
      {
        \tl_if_empty:cTF { g__enumext_resume_last_keys_ \int_to_roman:n {##1} _tl }
          {
            \__enumext_resume_counter:
          }
          {
            \__enumext_resume_counter:
            \exp_args:Ne \keys_set:nv
              { enumext / level-\int_use:N \l__enumext_level_int }
              { g__enumext_resume_last_keys_ \int_to_roman:n {##1} _tl }
          }
      }
    \int_compare:nNnT { \l__enumext_level_int } > { 0 }
      {
        \int_step_function:nN { \l__enumext_level_int } \__enumext_tmp:n
      }
    \int_compare:nNnT { \l__enumext_level_h_int } = { 1 }
      {
        \tl_if_empty:NTF \g__enumext_resume_last_keys_vii_tl
          {
            \__enumext_resume_counter:
          }
          {
            \__enumext_resume_counter:
            \keys_set:nV { enumext / enumext* } \g__enumext_resume_last_keys_vii_tl
          }
      }
  }
```

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

13.27 The `\resetenumext` command

Sometimes it is necessary to be able to reset the *"counters"* of the environments according to some value, for example `\chapter`. Since we use *"internal counters"* for the `resume` and `resume*` keys which set the *start value*, but are not accessible by the user, it is to provide a public command for this. This implementation is an adaptation of the answers given by Clea F. Rees (`@cfr`) and Jonathan P. Spratte (`@Skillmon`) in Correct implementation of optional argument (comma-separated) in expl3.

`\resetenumext`
`\__enumext_standard_reset:nn`
`\__enumext_starred_reset:n`
`\__enumext_reset_count_resume:nn`
`\__enumext_reset_count_resume:en`
`\__enumext_reset_count_resume_all:n`

The `\resetenumext` command *"resets"* the *start value* of the *"counters"* for the `enumext` and `enumext*` environments along with the *"internal counters"* used by the keys `resume` *without value* and `resume*` according to the value of {⟨*some counter*⟩}.

```
\NewDocumentCommand \resetenumext { s o m }
```

```
  {
    \bool_if:nTF {#1}
      {
        \__enumext_reset_count_resume_all:n {#3}
      }
      {
        \tl_if_novalue:nTF {#2}
          {
            \__enumext_reset_count_resume_levels:n {#3}
          }
          {
            \str_if_eq:nnTF {#2} { * }
              { \__enumext_starred_reset:n {#3} }
              {
                \bool_lazy_and:nnTF
                  { \int_compare_p:nNn {#2} > 0 }
                  { \int_compare_p:nNn {#2} < 5 }
                  { \__enumext_standard_reset:nn {#2} {#3} }
                  {
                    \msg_error:nne { enumext } { out-of-range } { \int_eval:n {#2} }
                  }
              }
          }
      }
  }
\cs_new_protected:Npn \__enumext_standard_reset:nn #1 %#2
  {
    \__enumext_reset_count_resume:en { \int_to_roman:n {#1} } %{#2}
  }
\cs_new_protected:Npn \__enumext_starred_reset:n #1
  {
    \__enumext_reset_count_resume:nn { vii } {#1}
  }
\cs_new_protected:Npn \__enumext_reset_count_resume:nn #1 #2
  {
    \counterwithin*{enumX#1}{#2}
    \counterwithin*{__enumext_resume_#1_int}{#2}
  }
\cs_generate_variant:Nn \__enumext_reset_count_resume:nn { e }
\cs_new_protected:Npn \__enumext_reset_count_resume_all:n #1
  {
    \clist_map_inline:nn { i,ii,iii,iv,vii }
      {
        \__enumext_reset_count_resume:nn { ##1 } { #1 }
      }
  }
\cs_new_protected:Npn \__enumext_reset_count_resume_levels:n #1
  {
    \clist_map_inline:nn { i,ii,iii,iv }
      {
        \__enumext_reset_count_resume:nn { ##1 } { #1 }
      }
  }
```

(*End of definition for* `\resetenumext` *and others. This function is documented on page* 11.)

13.28 The reset and reset* keys

The `\resetenumext` command does not work, for example, after an unnumbered chapter, so it is preferable to provide a pair of ⟨*keys*⟩ that adjust the internal variables if necessary.

reset
reset*

We define the keys `reset` and `reset*` for the *"all levels"* of `enumext` and `enumext*`.

```
\cs_set_protected:Npn \__enumext_tmp:n #1
  {
    \keys_define:nn { enumext / #1 }
      {
        reset  .code:n = \__enumext_standard_reset_key:,
        reset  .value_forbidden:n = true,
        reset* .code:n = \__enumext_standard_reset_key_star:,
        reset* .value_forbidden:n = true,
      }
  }
```

```
\clist_map_inline:nn {level-1, level-2, level-3, level-4} { \__enumext_tmp:n {#1} }
\keys_define:nn { enumext / enumext* }
  {
    reset  .code:n = \__enumext_starred_reset_key:,
    reset  .value_forbidden:n = true,
    reset* .code:n = \__enumext_starred_reset_key:,
    reset* .value_forbidden:n = true,
  }
```

(*End of definition for* `reset` *and* `reset*`.)

13.28.1 Internal functions for reset and reset* keys

__enumext_standard_reset_key:
__enumext_standard_reset_key_star:
__enumext_starred_reset_key:

The function `\__enumext_standard_reset_key:` will be handled by the `reset` key and will *"reset"* the counter `\c@__enumext_resume_X_int` to *"zero"* according to the *level* at which it is executed within the `enumext` environment.

```
\cs_new_protected:Nn \__enumext_standard_reset_key:
  {
    \int_compare:nNnT { \l__enumext_level_int } > { 0 }
      {
        \int_if_exist:cT { c@ __enumext_resume_ \int_to_roman:n { \l__enumext_level_int } _int }
          {
            \int_gzero:c { c@ __enumext_resume_ \int_to_roman:n { \l__enumext_level_int } _int }
          }
      }
  }
```

The function `\__enumext_standard_reset_key_star:` will be handled by the `reset*` key and will *"reset"* the counters `\c@__enumext_resume_X_int` to *"zero"* from the *level* at which it is executed within the `enumext` environment to the *lower levels*.

```
\cs_new_protected:Nn \__enumext_standard_reset_key_star:
  {
    \cs_set:Npn \__enumext_tmp:n ##1
      {
        \int_if_exist:cT { c@ __enumext_resume_ \int_to_roman:n {##1} _int }
          {
            \int_gzero:c { c@ __enumext_resume_ \int_to_roman:n {##1} _int }
          }
      }
    \int_compare:nNnT { \l__enumext_level_int } > { 0 }
      {
        \int_step_function:nnN { \l__enumext_level_int } { 4 } \__enumext_tmp:n
      }
  }
```

The function `\__enumext_starred_reset_key:` will be handled by `reset` keys and `reset*` will *"reset"* the counter `\c@__enumext_resume_vii_int` to *"zero"* when executed in the `enumext*` environment.

```
\cs_new_protected:Nn \__enumext_starred_reset_key:
  {
    \int_gzero:c { c@ __enumext_resume_vii_int }
  }
```

(*End of definition for* `\__enumext_standard_reset_key:`, `\__enumext_standard_reset_key_star:`, *and* `\__enumext_starred_reset_key:`.)

13.29 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.

13.29.1 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`.)

13.29.2 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 anskey*, keyans, keyans* and keyanspic environments and will set to "true" the variable \l__enumext_check_answers_bool used for internal checking answers mechanism set by the check-ans and no-store keys, copy {⟨*store name*⟩} into the variable \g__enumext_store_name_tl.

```
\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
```

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*.

```
    \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:`.)

13.29.3 The check answer mechanism

The internal mechanism for *"checking answers"* 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.

13.29.4 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`.)

13.29.5 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.*)

13.29.6 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`.)

13.30 Keys and functions associated with storage

13.30.1 Keys for `marks`, `wrap` and `show`

The enumext package provides a set of ⟨*keys*⟩ for manipulating *"symbol marks"* associated with *"answers"* and how they are displayed and stored in the *sequence* and *prop list* as well as an internal *"label and ref"* system.

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

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

```
\cs_set_protected:Npn \__enumext_tmp:nn #1 #2
  {
    \keys_define:nn { enumext / #1 }
      {
        mark-ans*  .tl_set:c  = { l__enumext_mark_answer_sym_#2_tl },
        mark-ans*  .initial:n = \textasteriskcentered,
        mark-ans*  .value_required:n = true,
        mark-pos*  .choice:,
        mark-pos* / left    .code:n = \str_set:cn { l__enumext_mark_position_#2_str } { l },
        mark-pos* / right   .code:n = \str_set:cn { l__enumext_mark_position_#2_str } { r },
        mark-pos* / center  .code:n = \str_set:cn { l__enumext_mark_position_#2_str } { c },
        mark-pos* / unknown .code:n =
                              \msg_error:nneee { enumext } { unknown-choice }
                                { mark-pos } { left,~right,~center } { \exp_not:n {##1} },
        mark-pos*  .initial:n = right,
        mark-pos*  .value_required:n = true,
        mark-sep*  .dim_set:c = { l__enumext_mark_sym_sep_#2_dim },
        mark-sep*  .value_required:n = true,
        wrap-ans*  .cs_set_protected:cp = { __enumext_keyans_wrapper_item_#2:n } ##1,
        wrap-ans*  .value_required:n = true,
        wrap-opt   .cs_set_protected:cp = { __enumext_keyans_wrapper_opt_#2:n } ##1,
```

```
        wrap-opt   .initial:n = [{##1}],
        wrap-opt   .value_required:n = true,
        save-sep   .tl_set:c = { l__enumext_store_keyans_item_opt_sep_#2_tl },
        save-sep   .initial:n = {,~},
        save-sep   .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}{v}, {keyans*}{viii} } { \__enumext_tmp:nn #1 }
```

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

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

We add the ⟨*keys*⟩ `mark-ref` and `save-ref` related to the *"storage system"* and internal mechanism of *"label and ref"* along with the ⟨*keys*⟩ `show-ans`, `show-pos` and the ⟨*keys*⟩ `mark-ans`, `mark-pos`, `mark-sep` and `wrap-ans` for the command `\anskey`, the environment `anskey*` and the the ⟨*keys*⟩ for environments `keyans` and `keyans*` only at the *first level* of `enumext` and `enumext*`.

```
\cs_set_protected:Npn \__enumext_tmp:n #1
  {
    \keys_define:nn { enumext / #1 }
      {
        mark-ref  .tl_set:N   = \l__enumext_mark_ref_sym_tl,
        mark-ref  .initial:n  = \textreferencemark,
        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,
        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-ans  .tl_set:N  = \l__enumext_mark_answer_sym_tl,
        mark-ans  .initial:n = \textasteriskcentered,
        mark-ans  .value_required:n = true,
        mark-sep  .dim_set:N  = \l__enumext_mark_sym_sep_dim,
        mark-sep  .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 / center  .code:n = \str_set:Nn \l__enumext_mark_position_str { c },
        mark-pos / unknown .code:n =
                             \msg_error:nneee { enumext } { unknown-choice }
                               { mark-pos } { left,~right,~center } { \exp_not:n {##1} },
        mark-pos  .initial:n = right,
        mark-pos  .value_required:n = true,

        wrap-ans  .cs_set_protected:Np = \__enumext_anskey_wrapper:n ##1,
        wrap-ans  .initial:n =
                    {
                      \fbox{\parbox[t]{\dimeval{\itemwidth -2\fboxsep -2\fboxrule}}{##1}}
                    },
        wrap-ans  .value_required:n = true,
        mark-ans* .code:n = {
                              \keys_set:nn { enumext / keyans  } { mark-ans* = {##1} }
                              \keys_set:nn { enumext / keyans* } { mark-ans* = {##1} }
                            },
        mark-ans* .value_required:n = true,
        mark-pos* .code:n = {
                              \keys_set:nn { enumext / keyans  } { mark-pos* = {##1} }
                              \keys_set:nn { enumext / keyans* } { mark-pos* = {##1} }
                            },
        mark-pos* .value_required:n = true,
        mark-sep* .code:n = {
                              \keys_set:nn { enumext / keyans  } { mark-sep* = {##1} }
                              \keys_set:nn { enumext / keyans* } { mark-sep* = {##1} }
```

```
                              },
        mark-sep* .value_required:n = true,
        wrap-ans* .code:n = {
                              \keys_set:nn { enumext / keyans  } { wrap-ans* = {##1} }
                              \keys_set:nn { enumext / keyans* } { wrap-ans* = {##1} }
                              },
        wrap-ans* .value_required:n = true,
        wrap-opt  .code:n = {
                              \keys_set:nn { enumext / keyans  } { wrap-opt = {##1} }
                              \keys_set:nn { enumext / keyans* } { wrap-opt = {##1} }
                              },
        wrap-opt  .value_required:n = true,
        save-sep  .code:n = {
                              \keys_set:nn { enumext / keyans  } { save-sep = {##1} }
                              \keys_set:nn { enumext / keyans* } { save-sep = {##1} }
                              },
        save-sep  .value_required:n = true,
      }
  }
\clist_map_inline:nn { level-1, enumext* } { \__enumext_tmp:n {#1} }
```

(*End of definition for* `mark-ref` *and others.*)

13.30.2 Storing structure of the environments

The idea behind *"storing structure"* 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 argument* passed to the levels of the environment in which it is executed and *"storing"* this in the *sequence*.

`\__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 the *"storing keys"* filtered from the *optional argument* 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_store_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`.)

13.30.3 Setting save-key key

Since this *"storing structure"* in the *sequence* established by the `save-ans` key when executing `\anskey` or `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 *"storing structure"* in the *sequence*.

`save-key`

The values set by this key passed in the *optional argument* 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`. Now 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 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.)

13.30.4 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 keys"* we want to *stored* in *sequence* where {#1} represents the *optional argument* 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 keys"* that are passed *"without value"* by excluding the resume, resume*, reset, reset*, no-store and base-fix keys.

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

The function __enumext_filter_save_key_pair:nn will be responsible for *"filtering keys"* that are passed *"with value"* by excluding the series, resume, save-ans, save-ref, save-key, check-ans, show-ans, save-pos, mark-ans, mark-pos, mark-sep, wrap-ans, mark-ans*, mark-pos*, mark-sep*, wrap-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  } {}
        { mark-ans  } {} { mark-pos   } {} { mark-sep    } {} { wrap-ans  } {}
        { mark-ans* } {} { mark-pos*  } {} { mark-sep*   } {} { wrap-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`.)

13.30.5 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 }
```

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

13.30.6 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 }
```

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

13.30.7 Functions for storing structure in the sequence

`\__enumext_store_level_open:`
`\__enumext_store_level_close:`

The *"storing structure"* 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 *"storing structure"* 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:`.)

13.30.8 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 }
    \hbox_overlap_left:n
      {
        \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`.)

13.31 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_not_p:n { \g__enumext_standar_bool } }
        { \bool_if_p:N \l__enumext_standar_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 \g__enumext_standar_bool }
        { \int_compare_p:nNn { \l__enumext_level_int } > { 0 } }
        { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
      }
      {
        \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 \g__enumext_standar_bool }
        { \bool_if_p:N \l__enumext_starred_bool }
        { \int_compare_p:nNn { \l__enumext_level_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:`.)

13.32 Common functions for `\anskey` and anskey* environment

`\__enumext_store_anskey_arg:n`

The internal function `\__enumext_store_anskey_arg: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_arg: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_arg: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 and wrap-sep keys.

```
\cs_new_protected:Npn \__enumext_anskey_show_wrap_arg:n #1
  {
    \par
    \bool_if:NTF \l__enumext_starred_bool
      {
        \dim_compare:nNnT { \l__enumext_mark_sym_sep_dim } = { \c_zero_dim }
          {
             \dim_set:Nn \l__enumext_mark_sym_sep_dim { \l__enumext_labelsep_vii_dim }
          }
        \__enumext_print_keyans_box:NN
          \l__enumext_labelwidth_vii_dim \l__enumext_mark_sym_sep_dim
      }
      {
        \dim_compare:nNnT { \l__enumext_mark_sym_sep_dim } = { \c_zero_dim }
          {
            \dim_set:Nn \l__enumext_mark_sym_sep_dim
              {
                \dim_use:c {l__enumext_labelsep_ \__enumext_level: _dim }
              }
          }
        \__enumext_print_keyans_box:cc
          { l__enumext_labelwidth_ \__enumext_level: _dim } { l__enumext_mark_sym_sep_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`.)

13.33 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*⟩`}`.

break-col
item-join
item-star
item-sym*
item-pos*
unknown
__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* `break-col` *and others.*)

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_arg: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_arg:n {#2}
            }
        }
    \group_end:
  }
```

(*End of definition for* `\anskey`. *This function is documented on page* 14.)

13.33.1 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:`.)

13.34 The environment anskey*

The original implementation of the `anskey*` environment used non-public functions from the `scontents`[4] package, which was not the best approach. Fortunately LaTeX release 2025-06-01 implemented the new `c-type` argument in the `ltcmd`[13], with which we can record the ⟨*body*⟩ of the environment in *verbatim mode* and, together with `\scantokens` do the work as the original implementation.

break-col
item-join
item-star
item-sym*
item-pos*
force-eol
write-env
overwrite
unknown

First we add the same keys from the \anskey command along with the force-eol, write-env and overwrite keys that were in the original implementation that used the scontents support package for these.

```
\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,
    force-eol .bool_set:N = \l__enumext_anskey_env_force_eol_bool,
    force-eol .initial:n  = false,
    force-eol .default:n  = true,
    write-env .code:n     = {
                              \bool_set_true:N \l__enumext_write_anskey_env_bool
                              \tl_set:Nn \l__enumext_write_anskey_env_file_name_tl {#1}
                            },
    write-env .value_required:n = true,
    overwrite .bool_set:N = \l__enumext_anskey_env_overwrite_bool,
    overwrite .initial:n  = false,
    overwrite .default:n  = true,
    unknown   .code:n     = { \__enumext_anskey_env_unknown:n {#1} },
  }
```

(*End of definition for* `break-col` *and others.*)

__enumext_anskey_env_unknown:n
__enumext_anskey_env_unknown:nn

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}
      }
  }
```

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

__enumext_anskey_env_file_if_writable:n
__enumext_anskey_env_file_if_writable:nT
__enumext_anskey_env_file_if_writable:nF
__enumext_anskey_env_file_if_writable:nTF

The conditional function __enumext_anskey_env_file_if_writable:n used by the write-env and overwrite keys in the anskey* environment to determine whether the output file is written or overwritten.

```
\prg_new_protected_conditional:Npnn \__enumext_anskey_env_file_if_writable:n #1 { T, F, TF }
  {
    \bool_if:NTF \l__enumext_write_anskey_env_bool
      {
        \file_if_exist:nTF {#1}
          {
            \bool_if:NTF \l__enumext_anskey_env_overwrite_bool
              {
                \msg_warning:nne { enumext } { overwrite-file } {#1}
                \prg_return_true:
              }
              {
                \msg_warning:nne { enumext } { not-writing } {#1}
                \prg_return_false:
              }
          }
          {
```

```
            \msg_warning:nne { enumext } { writing-file } {#1}
            \prg_return_true:
          }
      }
      { \prg_return_false: }
  }
```

The `\__enumext_anskey_env_file_write:nn` function is used by the `write-env` key in the `anskey*` environment to write the output file with the ⟨*body*⟩ of the environment.

```
\cs_new_protected:Npn \__enumext_anskey_env_file_write:nn #1#2
  {
    \__enumext_anskey_env_file_if_writable:nT {#1}
      {
        \iow_open:Nn \l__enumext_write_anskey_env_file_iow {#1}
        \iow_now:Nn  \l__enumext_write_anskey_env_file_iow {#2}
        \iow_close:N \l__enumext_write_anskey_env_file_iow
      }
  }
\cs_generate_variant:Nn \__enumext_anskey_env_file_write:nn { VV }
```

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

`anskey*` First, we'll call the function `\__enumext_anskey_env_safe_outer:` to make sure where we're running the environment, then, we'll check the state of the variable `\l__enumext_check_answers_bool` set by the key `no-store`. If it's true, we'll look for `[`⟨*key* = *val*⟩`]` and verify that the *argument* `c` ⟨*body*⟩ is not empty. Finally, we'll run the internal check function `\__enumext_anskey_env_safe_inner:n` and call the function `\__enumext_store_anskey_arg:n`.

```
\NewDocumentEnvironment{anskey*}{ o c }
  {
    \__enumext_anskey_env_safe_outer:
    \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_env_safe_inner:
            \__enumext_store_anskey_env:n {#2}
          }
      }
  } { }
```

(*End of definition for* `anskey*`. *This function is documented on page 15.*)

13.34.1 Internal functions for the environment

`\__enumext_anskey_env_safe_outer:` `\__enumext_anskey_env_safe_inner:` `\__enumext_store_anskey_env:n` The function `\__enumext_store_anskey_safe_outer:` will return the appropriate messages when `anskey*` is executed outside the environment in which the `save-ans` key was activated or within the `keyans`, `keyans*` or `keyanspic` environments.

```
\cs_new_protected:Nn \__enumext_anskey_env_safe_outer:
  {
    \bool_if:NF \l__enumext_store_active_bool
      {
        \msg_error:nnn { enumext } { anskey-env-error } { 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 }
      }
  }
```

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

```
\cs_new_protected:Nn \__enumext_anskey_env_safe_inner:
  {
    \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* }
      }
  }
```

The __enumext_store_anskey_env:n function will first pass the argument c ⟨*body*⟩ to the variable \l__enumext_store_anskey_env_tl and replace the macro \obeyedline with ^^J and then execute the write-env and overwrite keys, check the state of the variable \l__enumext_anskey_env_force_eol_bool managed by the force-eol key and we will add \c__enumext_anskey_env_hidden_space_str if necessary. Finally we will use \exp_args:Ne on the __enumext_store_anskey_arg:n to expand the __enumext_scan_tokens:n function which rescans the \l__enumext_store_anskey_env_tl variable before processing it.

```
\cs_new_protected:Npn \__enumext_store_anskey_env:n #1
  {
    \tl_set:Nn \l__enumext_store_anskey_env_tl {#1}
    \RenewDocumentCommand \obeyedline { } { \iow_char:N \^^J }
    \tl_replace_all:Nee \l__enumext_store_anskey_env_tl { \obeyedline } { \iow_char:N \^^J }
    \__enumext_anskey_env_file_write:VV
      \l__enumext_write_anskey_env_file_name_tl \l__enumext_store_anskey_env_tl
    \bool_if:NF \l__enumext_anskey_env_force_eol_bool
      {
        \tl_put_right:Ne \l__enumext_store_anskey_env_tl
          {
            \c__enumext_anskey_env_hidden_space_str
          }
      }
    \exp_args:Ne
      \__enumext_store_anskey_arg:n
        {
          \__enumext_scan_tokens:n { \l__enumext_store_anskey_env_tl }
        }
  }
```

Since \obeyedline can be redefined by the user, for example to \mbox{}\par, it is necessary to redefine it to ^^J in order to use \tl_replace_all:Nee otherwise it returns an error.

(*End of definition for* __enumext_anskey_env_safe_outer:, __enumext_anskey_env_safe_inner:, *and* __enumext_store_anskey_env:n.)

13.35 Executing check-ans system 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 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:
              }
```

```
3256                { \__enumext_check_ans_log: }
3257            }
3258          \__enumext_reset_global_vars:
3259        }
3260    }
```

This function is passed to the function `\__enumext_after_env:nn` for the environments `enumext` (§13.42) and `enumext*` (§13.47) and it is executed only when the environments are not nested or at some level of these..

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

13.36 Common functions for keyans, keyans* and keyanspic

13.36.1 Storing content in prop list

`\__enumext_keyans_addto_prop:n`

The function `\__enumext_keyans_addto_prop:n` will pass the the current ⟨*label*⟩ for `\item*` in `keyans` environment and the current ⟨*label*⟩ for `\anspic*` in `keyanspic` environment followed by the ⟨*contents*⟩ of the *optional argument* of both commands to the `\l__enumext_store_current_label_tl` variable, which will be stored to the *prop list* defined by the `save-ans` key using the function `\__enumext_store_addto_prop:V`.

```
3261 \cs_new_protected:Npn \__enumext_keyans_addto_prop:n #1
3262   {
3263     \tl_clear:N \l__enumext_store_current_label_tl
3264     \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
3265       {
3266         \tl_put_right:Ne \l__enumext_store_current_label_tl { \l__enumext_label_vi_tl }
3267       }
3268       {
3269         \tl_put_right:Ne \l__enumext_store_current_label_tl { \l__enumext_label_v_tl }
3270       }
```

If the *optional argument* is present and the `save-sep` key is not empty, we save it.

```
3271     \tl_if_novalue:nF { #1 }
3272       {
3273         \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_v_tl
3274           {
3275             \tl_put_right:NV \l__enumext_store_current_label_tl \l__enumext_store_keyans_item_opt
3276           }
3277         \tl_put_right:Nn \l__enumext_store_current_label_tl { #1 }
3278       }
3279     \__enumext_store_addto_prop:V \l__enumext_store_current_label_tl
3280   }
```

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

13.36.2 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 `\anskey` basically because in this environments the interest is in the current ⟨*label*⟩ for `\item*` and `\anspic*` with the ⟨*contents*⟩ of the *optional argument*. 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 references.

```
3281 \cs_new_protected:Nn \__enumext_keyans_store_ref:
3282   {
3283     \bool_if:NT \l__enumext_store_ref_key_bool
3284       {
3285         \cs_set_protected:Npn \__enumext_tmp:n ##1
3286           {
3287             \tl_set_eq:cc { l__enumext_label_copy_##1_tl } { l__enumext_label_##1_tl }
3288             \tl_reverse:c { l__enumext_label_copy_##1_tl }
3289             \tl_remove_once:cn { l__enumext_label_copy_##1_tl } { . }
3290             \tl_reverse:c { l__enumext_label_copy_##1_tl }
3291           }
3292         \clist_map_inline:nn { i, v, vi, vii, viii } { \__enumext_tmp:n {##1} }
3293         \__enumext_keyans_store_ref_aux_i:
3294       }
3295   }
```

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:`.)

13.36.3 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_v_tl
          {
            \tl_put_right:NV \l__enumext_store_current_label_tl \l__enumext_store_keyans_item_opt
          }
        \tl_put_right:Nn \l__enumext_store_current_label_tl { #1 }
```

```
3351      }
3352    \__enumext_keyans_addto_seq_link:
3353  }
```

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.

```
3354 \cs_new_protected:Nn \__enumext_keyans_addto_seq_link:
3355  {
3356    \bool_lazy_and:nnT
3357      { \bool_if_p:N \l__enumext_store_ref_key_bool }
3358      { \bool_if_p:N \l__enumext_hyperref_bool }
3359      {
3360        \tl_put_right:Ne \l__enumext_store_current_label_tl
3361          {
3362            \hfill \exp_not:N \hyperlink
3363              {
3364                \exp_not:V \l__enumext_newlabel_arg_one_tl
3365              }
3366              { \exp_not:V \l__enumext_mark_ref_sym_tl }
3367          }
3368      }
3369    \__enumext_store_addto_seq:V \l__enumext_store_current_label_tl
3370    \bool_if:NT \l__enumext_check_answers_bool
3371      {
3372        \int_gincr:N \g__enumext_item_anskey_int
3373      }
3374  }
```

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

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

__enumext_keyans_save_item_opt:n
__enumext_keyans_show_item_opt:
__enumext_keyans_show_item_opt_viii:

The function __enumext_keyans_save_item_opt:n will save the optional argument of \item* and \anspic* in the variable \l__enumext_store_current_opt_arg_tl.

```
3375 \cs_new_protected:Npn \__enumext_keyans_save_item_opt:n #1
3376  {
3377    \tl_if_novalue:nF { #1 }
3378      {
3379        \tl_set:Nn \l__enumext_store_current_opt_arg_tl { #1 }
3380      }
3381  }
```

The function __enumext_keyans_show_item_opt: will print the optional arguments of \item* and \anspic* when the show-ans or show-pos keys are set next to the key wrap-opt in keyans and keyanspic environments.

```
3382 \cs_new_protected:Nn \__enumext_keyans_show_item_opt:
3383  {
3384    \tl_if_empty:NF \l__enumext_store_current_opt_arg_tl
3385      {
3386        \bool_lazy_or:nnT
3387          { \bool_if_p:N \l__enumext_show_answer_bool }
3388          { \bool_if_p:N \l__enumext_show_position_bool }
3389          {
3390            \__enumext_keyans_wrapper_opt_v:n
3391              { \l__enumext_store_current_opt_arg_tl } \c_space_tl
3392          }
3393      }
3394  }
```

The function __enumext_keyans_show_item_opt_viii: will print the optional argument of \item* when the show-ans or show-pos keys are set next to the key wrap-opt in keyans* environment.

```
3395 \cs_new_protected:Nn \__enumext_keyans_show_item_opt_viii:
3396  {
3397    \tl_if_empty:NF \l__enumext_store_current_opt_arg_tl
3398      {
3399        \bool_lazy_or:nnT
3400          { \bool_if_p:N \l__enumext_show_answer_bool }
3401          { \bool_if_p:N \l__enumext_show_position_bool }
3402          {
```

```
            \__enumext_keyans_wrapper_opt_viii:n
              { \l__enumext_store_current_opt_arg_tl } \c_space_tl
          }
      }
  }
```

(*End of definition for* `\__enumext_keyans_save_item_opt:n`, `\__enumext_keyans_show_item_opt:`, *and* `\__enumext_keyans_show_item_opt_viii:`.)

`\__enumext_keyans_pos_mark_set:`
`\__enumext_keyans_show_ans:`
`\__enumext_keyans_show_pos:`

The function `\__enumext_keyans_pos_mark_set:` adjusts the horizontal spaces for the `mark-sep*` key taking into account the value of the `align` key and the width of ⟨*label*⟩.

```
\cs_new_protected:Nn \__enumext_keyans_pos_mark_set:
  {
    \__enumext_label_width_by_box:Nn
      \l__enumext_mark_sep_tmpa_dim { \l__enumext_label_v_tl }
    \str_case:Vn \l__enumext_align_label_pos_v_str
      {
        { l }
            {
              \dim_set:Nn \l__enumext_mark_sep_tmpb_dim { \c_zero_dim }
            }
        { r }
            {
              \dim_set:Nn \l__enumext_mark_sep_tmpb_dim
                { \l__enumext_labelwidth_v_dim - \l__enumext_mark_sep_tmpa_dim }
            }
        { c }
            {
              \dim_set:Nn \l__enumext_mark_sep_tmpb_dim
                { 0.5\l__enumext_labelwidth_v_dim - 0.5\l__enumext_mark_sep_tmpa_dim }
            }
       }
```

Here we set the default values for the key `mark-ans*`, `mark-sep*` and `mark-pos*`.

```
    \dim_compare:nNnT { \l__enumext_mark_sym_sep_v_dim } = { \c_zero_dim }
      {
        \dim_set:Nn \l__enumext_mark_sym_sep_v_dim { \l__enumext_labelsep_v_dim }
      }
    \tl_set_eq:NN \l__enumext_mark_answer_sym_tl \l__enumext_mark_answer_sym_v_tl
    \dim_add:Nn \l__enumext_mark_sym_sep_v_dim { \l__enumext_mark_sep_tmpb_dim }
    \str_set_eq:NN \l__enumext_mark_position_str \l__enumext_mark_position_v_str
  }
```

The function `\__enumext_keyans_show_ans:` will print the ⟨*symbol*⟩ set by the `mark-ans*` key when the `show-ans` key is active.

```
\cs_new_protected:Nn \__enumext_keyans_show_ans:
  {
    \bool_lazy_all:nT
      {
        { \bool_if_p:N \l__enumext_show_answer_bool   }
        { \bool_if_p:N \l__enumext_item_wrap_key_bool }
      }
      {
        \__enumext_keyans_pos_mark_set:
        \__enumext_print_keyans_box:NN
          \l__enumext_labelwidth_v_dim \l__enumext_mark_sym_sep_v_dim
      }
  }
```

The function `\__enumext_keyans_show_pos:` will print the ⟨*position*⟩ of the stored content in *prop list*. Need add `1` to `\g__enumext_`⟨*store name*⟩`_prop` for `keyans` environment.

```
\cs_new_protected:Nn \__enumext_keyans_show_pos:
  {
    \int_compare:nNnTF { \l__enumext_keyans_level_int } = { 1 }
      {
        \int_incr:N \l__enumext_show_pos_tmp_int
      }
      {
        \int_zero:N \l__enumext_show_pos_tmp_int
      }
    \bool_lazy_all:nT
      {
```

```
      { \bool_if_p:N \l__enumext_show_position_bool }
      { \bool_if_p:N \l__enumext_item_wrap_key_bool }
    }
    {
      \tl_set:Ne \l__enumext_mark_answer_sym_v_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 }
                + \l__enumext_show_pos_tmp_int
              }
            ]
          \group_end:
        }
      \__enumext_keyans_pos_mark_set:
      \__enumext_print_keyans_box:NN
        \l__enumext_labelwidth_v_dim \l__enumext_mark_sym_sep_v_dim
    }
}
```

(*End of definition for* `\__enumext_keyans_pos_mark_set:`, `\__enumext_keyans_show_ans:`, *and* `\__enumext_keyans_show_pos:`.)

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

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

When *labeling* PDF is active `\makelabel` is redefined as `\hss #1` and the only way to get the `align` key to work correctly is to redefine `\makelabel` using `\makebox`. The best way to implement this is to use the conditional command `\IfDocumentMetadataTF` to force this redefinition and the dedicated `mode-box` key to manually activate it by the user.

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

`\__enumext_default_item:n`

First we will see if the *optional argument* is present, if it is NOT present we will check the state of the variable `\l__enumext_check_answers_bool` set by the key `no-store`, set the boolean variable `\l__enumext_wrap_label_X_bool` to "true" for the key `wrap-label` and execute `\__enumext_item_std:w` and the key `itemindent`, otherwise we will check the state of the boolean variable `\l__enumext_wrap_label_opt_X_bool` set by the key `wrap-label*` and execute `\__enumext_item_std:w` with the *optional argument* and the key `itemindent`.

```
\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_item_starred_exec:nn`
`\__enumext_item_starred_exec:`

The `\item*`, `\item*[`⟨*symbol*⟩`]` and `\item*[`⟨*symbol*⟩`][`⟨*offset*⟩`]` works like the *numbered* `\item`, but placing a ⟨*symbol*⟩ to the *"left"* of the ⟨*label*⟩ separated from it by the value the second *optional argument* ⟨*offset*⟩.

#1 : `\l__enumext_item_symbol_X_tl`
#2 : `\l__enumext_item_symbol_sep_X_dim`

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

```
\cs_new_protected:Npn \__enumext_item_starred_exec:nn #1 #2
  {
    \tl_if_novalue:nTF {#1}
      {
        \tl_gset_eq:Nc
          \g__enumext_item_symbol_aux_tl { l__enumext_item_symbol_ \__enumext_level: _tl }
      }
      {
        \tl_gset:Nn \g__enumext_item_symbol_aux_tl {#1}
      }
    \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 }
  }
```

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

```
\cs_new_protected:Nn \__enumext_item_starred_exec:
  {
    \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 } }
        \hbox_overlap_left:n { \g__enumext_item_symbol_aux_tl }
        \skip_horizontal:n { \dim_use:c { l__enumext_item_symbol_sep_ \__enumext_level: _dim } }
      }
  }
```

(*End of definition for* __enumext_item_starred_exec:nn *and* __enumext_item_starred_exec:.)

__enumext_redefine_item:

The function __enumext_redefine_item: will redefine the \item command in the enumext environment adding \item*. This function are passed to __enumext_list_arg_two_X: used in the definition of the enumext environment (§13.42).

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

(*End of definition for* __enumext_redefine_item:.)

__enumext_make_label:
__enumext_make_label_std:
__enumext_make_label_box:

The function __enumext_make_label: redefine \makelabel for the keys mode-box, align, font, wrap-label, wrap-label* and \item* for enumext environment. This function are passed to __enumext_list_arg_two_X: used in the definition of the enumext environment (§13.42).

```
\cs_new_protected:Nn \__enumext_make_label:
```

```
  {
    \IfDocumentMetadataTF
      {
        \__enumext_make_label_box:
      }
      {
        \bool_if:NTF \l__enumext_mode_box_bool
          {
             \__enumext_make_label_box:
          }
          {
             \__enumext_make_label_std:
          }
      }
  }
```

Standard definition when `\DocumentMetadata` is not active.

```
\cs_new_protected:Nn \__enumext_make_label_std:
  {
    \RenewDocumentCommand \makelabel { m }
      {
        \tl_use:c { l__enumext_label_fill_left_ \__enumext_level: _tl }
        \__enumext_item_starred_exec:
        \tl_use:c { l__enumext_label_font_style_ \__enumext_level: _tl }
        \bool_if:cTF { l__enumext_wrap_label_ \__enumext_level: _bool }
          {
            \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_aux_tl
      }
  }
```

Definition using `\makebox` when `\DocumentMetadata` is active or `mode-box` is active.

Here it is necessary to use `\strut\smash` to maintain text *alignment* in case the user wants to use `\labelbx` for example. In my experiments with *mimicking* the `description` environment it was the only way out and it seems to have no adverse effects and may serve in the future as a basis for a more generic `list` environment package than enumext.

```
\cs_new_protected:Nn \__enumext_make_label_box:
  {
    \RenewDocumentCommand \makelabel { m }
      {
        \strut\smash
          {
            \makebox
              [ \dim_use:c { l__enumext_labelwidth_ \__enumext_level: _dim } ]
              [ \str_use:c { l__enumext_align_label_pos_ \__enumext_level: _str } ]
              {
                \__enumext_item_starred_exec:
                \tl_use:c { l__enumext_label_font_style_ \__enumext_level: _tl }
                \bool_if:cTF { l__enumext_wrap_label_ \__enumext_level: _bool }
                  {
                    \use:c { __enumext_wrapper_label_ \__enumext_level: :n } { ##1 }
                  }
                  { ##1 }
                \tl_gclear:N \g__enumext_item_symbol_aux_tl
              }
          } % close smash
      }
  }
```

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

13.38 Setting `item-sym*` and `item-pos*` keys

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

`item-sym*` `item-pos*`

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

```
\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  = {\textborn},
        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*`.)

13.39 Handling unknown keys

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

Well, the paragraph above is not so real, after all I had to add more ⟨*keys*⟩ than I had planned, not everything turns out the way one thinks in life.

13.39.1 Handling unknown keys for keyans, keyans* and keyanspic

unknown
__enumext_keyans_unknown_keys:n
__enumext_keyans_unknown_keys:nn

Define and set unknown key for keyans, keyans* and keyanspic environments. Here it is necessary to set `\l__enumext_envir_name_tl` in case an unknown key is passed using `\setenumext`.

```
\cs_set_protected:Npn \__enumext_tmp:n #1
  {
    \keys_define:nn { enumext / #1 }
      {
        unknown .code:n = {
                            \tl_set:Nn \l__enumext_envir_name_tl {#1}
                            \__enumext_keyans_unknown_keys:n {##1}
                          },
      }
  }
\clist_map_inline:nn { keyans, keyans*, keyanspic } { \__enumext_tmp:n {#1} }
```

Internal functions for handling unknown key.

```
\cs_new_protected:Npn \__enumext_keyans_unknown_keys:n #1
  {
    \exp_args:NV \__enumext_keyans_unknown_keys:nn \l_keys_key_str {#1}
  }
\cs_new_protected:Npn \__enumext_keyans_unknown_keys:nn #1#2
  {
    \tl_if_blank:nTF {#2}
      {
        \msg_error:nne { enumext } { keyans-unknown-key } {#1}
      }
      {
        \msg_error:nnee { enumext } { keyans-unknown-key-value } {#1} {#2}
      }
  }
```

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

13.39.2 Handling unknown keys for enumext*

unknown
__enumext_starred_unknown_keys:n
__enumext_starred_unknown_keys:nn

Define and set unknown key for enumext* environment.

```
\keys_define:nn { enumext / enumext* }
  {
    unknown .code:n = {
                        \tl_set:Nn \l__enumext_envir_name_tl { enumext* }
                        \__enumext_starred_unknown_keys:n {#1}
                      },
  }
```

Internal functions for handling unknown key.

```
\cs_new_protected:Npn \__enumext_starred_unknown_keys:n #1
  {
    \exp_args:NV \__enumext_starred_unknown_keys:nn \l_keys_key_str {#1}
  }
```

```
\cs_new_protected:Npn \__enumext_starred_unknown_keys:nn #1#2
  {
    \tl_if_blank:nTF {#2}
      {
        \msg_error:nne { enumext } { starred-unknown-key } {#1}
      }
      {
        \msg_error:nnee { enumext } { starred-unknown-key-value } {#1} {#2}
      }
  }
```

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

13.39.3 Handling unknown keys for enumext

unknown

Defines and set the key unknown for enumext environment.

__enumext_standar_unknown_keys:n
__enumext_standar_unknown_keys:nn

```
\cs_set_protected:Npn \__enumext_tmp:n #1
  {
    \keys_define:nn { enumext / level-#1 }
      {
        unknown .code:n = {
                            \int_set:Nn \l__enumext_level_int { #1 }
                            \tl_set:Nn \l__enumext_envir_name_tl { enumext }
                            \__enumext_standar_unknown_keys:n {##1}
                          },
      }
  }
\clist_map_inline:nn {1, 2, 3, 4} { \__enumext_tmp:n {#1} }
```

Internal functions for handling unknown key.

```
\cs_new_protected:Npn \__enumext_standar_unknown_keys:n #1
  {
    \exp_args:NV \__enumext_standar_unknown_keys:nn \l_keys_key_str {#1}
  }
\cs_new_protected:Npn \__enumext_standar_unknown_keys:nn #1#2
  {
    \tl_if_blank:nTF {#2}
      {
        \msg_error:nne { enumext } { standar-unknown-key } {#1}
      }
      {
        \msg_error:nnee { enumext } { standar-unknown-key-value } {#1} {#2}
      }
  }
```

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

13.40 Redefining \item and \makelabel in keyans

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

__enumext_keyans_default_item:n

The function __enumext_keyans_default_item:n executes the original behavior of the \item along with the keys wrap-label, wrap-label* and itemindent.

```
\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 will take as argument #1 the *optional argument* [⟨*content*⟩] passed to \item* and save it via the __enumext_keyans_save_item_opt:n function, then activate the wrap-label key, execute \item using __enumext_item_std:w, the itemindent key and print the *optional argument* using the __enumext_keyans_show_item_opt: function handled by the wrap-opt key.

```
\cs_new_protected:Npn \__enumext_keyans_starred_item:n #1
  {
    \__enumext_keyans_save_item_opt: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_opt:
```

Now *store* the current ⟨*label*⟩ first in the *prop list* (including the *optional argument*), run the internal *"label and ref"* system if the save-ref key is active, then *store* in the *sequence* and finally increments \g__enumext_check_starred_cmd_int for internal check system.

```
    \__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 argument* and *optional argument* by the __enumext_list_arg_two_v: function in the definition of the keyans environment. Here we will set to true the variable \l__enumext_item_wrap_key_bool used by the wrap-ans* key only when \item* is executed and additionally we need to use \peek_remove_spaces:n to avoid an unwanted space when using \item* together with the itemindent key. This function are passed to __enumext_list_arg_two_v: used in the definition of the keyans environment (§13.41).

```
\cs_new_protected:Nn \__enumext_keyans_redefine_item:
  {
    \RenewDocumentCommand \item { s o }
      {
        \bool_if:nTF {##1}
          {
            \bool_set_true:N \l__enumext_item_wrap_key_bool % wrap-ans*
            \peek_remove_spaces:n
              {
                \__enumext_keyans_starred_item:n {##2}
              }
          }
          {
            \bool_set_false:N \l__enumext_item_wrap_key_bool
            \__enumext_keyans_default_item:n {##2}
          }
      }
  }
```

(*End of definition for* \item* *and* __enumext_keyans_redefine_item:. *This function is documented on page* 17.)

__enumext_keyans_make_label:
__enumext_keyans_wrapper_label:n
__enumext_keyans_make_label_std:
__enumext_keyans_make_label_box:

The function __enumext_keyans_make_label: redefine \makelabel for the keys mode-box, align, font, wrap-label, wrap-label*, wrap-ans* and \item* for keyans environment. This function are passed to __enumext_list_arg_two_v: used in the definition of the keyans environment (§13.41).

```
\cs_new_protected:Nn \__enumext_keyans_make_label:
  {
    \IfDocumentMetadataTF
      {
        \__enumext_keyans_make_label_box:
      }
      {
        \bool_if:NTF \l__enumext_mode_box_bool
          {
            \__enumext_keyans_make_label_box:
          }
          {
            \__enumext_keyans_make_label_std:
          }
      }
  }
```

We added conditionals to the __enumext_keyans_wraper_label:n function to handle the keys wrap-ans*, wrap-label and wrap-label*.

```
\cs_new_protected:Npn \__enumext_keyans_wrapper_label:n #1
  {
    \bool_lazy_all:nT
      {
        { \bool_if_p:N \l__enumext_wrap_label_v_bool         }
        { \bool_if_p:N \l__enumext_show_answer_bool          }
        { \bool_if_p:N \l__enumext_item_wrap_key_bool        }
        { \cs_if_exist_p:N \__enumext_keyans_wrapper_item_v:n }
      }
      {
        \cs_set_eq:NN \__enumext_wrapper_label_v:n \__enumext_keyans_wrapper_item_v:n
      }
    \bool_if:NTF \l__enumext_wrap_label_v_bool
      {
        \__enumext_wrapper_label_v:n { #1 }
      }
      { #1 }
  }
```

Standard definition when \DocumentMetadata is not active.

```
\cs_new_protected:Nn \__enumext_keyans_make_label_std:
  {
    \RenewDocumentCommand \makelabel { m }
      {
        \tl_use:N \l__enumext_label_fill_left_v_tl
        \__enumext_keyans_show_ans:
        \__enumext_keyans_show_pos:
        \tl_use:N \l__enumext_label_font_style_v_tl
        \__enumext_keyans_wrapper_label:n { ##1 }
        \tl_use:N \l__enumext_label_fill_right_v_tl
      }
  }
```

Definition using \makebox when \DocumentMetadata is active or mode-box is active.

```
\cs_new_protected:Nn \__enumext_keyans_make_label_box:
  {
    \RenewDocumentCommand \makelabel { m }
      {
        \strut\smash
          {
            \makebox[ \l__enumext_labelwidth_v_dim ][ \l__enumext_align_label_pos_v_str ]
              {
                \__enumext_keyans_show_ans:
                \__enumext_keyans_show_pos:
                \tl_use:N \l__enumext_label_font_style_v_tl
                \__enumext_keyans_wrapper_label:n { ##1 }
              }
          }
      }
  }
```

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

13.41 Second argument of the lists

At this point in the code we have already programmed most of the tools needed to create a *custom* list environment, remember that the __enumext_start_list:nn function takes two arguments, we have the "first" one ready, the "second" one we will define for all levels of the enumext environment, the keyans environment and the enumext* and keyans* environments.

Here we will implement the __enumext_list_arg_two_X: function, which will be responsible for setting all the list parameters, the counter, the redefinition of \item, \makelabel along with the keys ref, itemindent and show-length.

In the functions __enumext_list_arg_two_X: we will implement the *"counter"* for the environments, but we do NOT set the *"start value"* for it to be compatible with *tagged* PDF that should be done later.

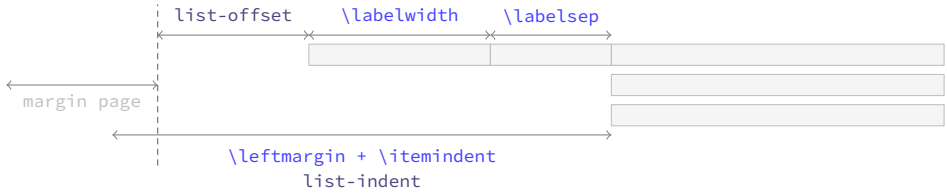


Figure 9: Representation of standard horizontal lengths in `list` environment.

13.41.1 Calculation of `\leftmargin` and `\itemindent`

Consider the figure 9 where the default margins (on the left) of a list are represented.
The idea is to have control over these margins so that our list does not overlap the left margin of the page. The *key* relationship is that the *"right edge"* of the `\labelsep` equals the *"right edge"* of the `\itemindent`, so that the left edge of the *"label box"* is at `\leftmargin`+`\itemindent` minus `\labelwidth`+`\labelsep`. Thus, the handling of the margins by the package will be as shown in the figure 10.

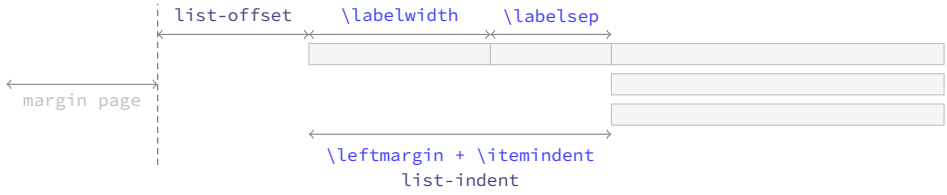


Figure 10: Representation of horizontal lengths concept in list in enumext.

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

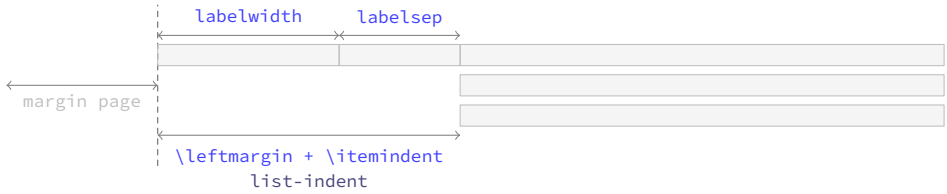


Figure 11: Default horizontal lengths in enumext.

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

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

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

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

```
\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`.)

13.41.2 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 } }
        \clist_map_inline:nn { beginparpenalty, itempenalty, endparpenalty }
          { \int_set_eq:cc {@####1} { l__enumext_####1_#1_int } }
        \usecounter { enumX#1 }
        \str_if_eq:nnTF {#1} { v }
          {
            \__enumext_keyans_redefine_item:
            \__enumext_keyans_make_label:
            \__enumext_keyans_ref:
            \__enumext_keyans_fake_item_indent:
            \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_indent:
            \bool_if:cT { l__enumext_show_length_#1_bool }
              {
                \msg_term:nnne { enumext } { list-lengths } {#1}
                  { \int_use:N \l__enumext_level_int }
              }
          }
      }
  }
\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 } }
      \clist_map_inline:nn { beginparpenalty, itempenalty, endparpenalty }
        { \int_set_eq:cc {@####1} { l__enumext_####1_#1_int } }
      \skip_set_eq:Nc \parsep  { l__enumext_itemsep_#1_skip }
      \skip_zero:N \partopsep
      \usecounter { enumX#1 }
      \__enumext_starred_ref:
      \str_if_eq:nnTF {#1} { vii }
        {
          \__enumext_fake_item_indent_vii:
          \bool_if:cT { l__enumext_show_length_vii_bool }
            { \msg_term:nnnn { enumext } { list-lengths-not-nested } { vii } { enumext* } }
        }
        {
          \__enumext_fake_item_indent_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:`.)

13.42 The environment enumext

`\__enumext_safe_exec:`

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

```
\cs_new_protected:Nn \__enumext_safe_exec:
  {
    \__enumext_is_not_nested:
    \__enumext_internal_mini_page:
    \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
    \bool_set_false:N \l__enumext_starred_bool
    \__enumext_is_on_first_level:
  }
```

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

`\__enumext_parse_keys:n`

The `\__enumext_parse_store_keys:n` function first we will clear the variable `\l__enumext_series_name_str` used by the key `series` and then we check if we are at the *"first level"*, if so we process the ⟨*keys*⟩ and then execute the function `\__enumext_parse_series:n` used by the key `series` and call the function `\__enumext_nested_base_line_fix:` used by the key `base-fix`, otherwise we will pass the ⟨*keys*⟩ to the inner levels of the environment then we execute the function `\__enumext_store_active_keys:n` and reprocess the ⟨*keys*⟩ to pass them to the *sequence* if the key `save-key` is not active.

```
\cs_new_protected:Npn \__enumext_parse_keys:n #1
  {
    \tl_if_novalue:nF {#1}
      {
        \str_clear:N \l__enumext_series_name_str
        \int_compare:nNnTF { \l__enumext_level_int } = { 1 }
          {
            \keys_set:nn { enumext / level-1 } {#1}
            \bool_if:NF \l__enumext_print_keyans_cmd_bool
```

```
                {
                  \__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}
              \bool_if:NF \l__enumext_print_keyans_cmd_bool
                {
                  \__enumext_parse_series:n {#1}
                }
            }
          \__enumext_store_active_keys:n {#1}
        }
    }
```

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

`\__enumext_start_store_level:` The `\__enumext_start_store_level:` function activate the *"storing structure"* mechanism in the *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 *"storing 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:
          }
      }
  }
```

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

`\__enumext_stop_store_level:` The `\__enumext_stop_store_level:` function stop the *"storing structure"* mechanism in the *sequence* for the command `\anskey` and the environment `anskey*`.

```
\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_stop_store_level:`.)

`\__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  }
        \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 }
          {
            \skip_zero:N \multicolsep
            \__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 and apply our *"vertical adjust"* spacing. For compatibility with *tagged* PDF, the closing of the `list` environment is executed here along with `\__enumext_stop_store_level:`.

```
\cs_new_protected:Nn \__enumext_multicols_stop:
  {
    \int_compare:nNnTF
      { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } > { 1 }
      {
        \__enumext_stop_list:
        \__enumext_stop_store_level:
        \end{multicols}
        \__enumext_unskip_unkern:
        \__enumext_unskip_unkern:
        \par\addvspace{ \skip_use:c { l__enumext_multicols_below_ \__enumext_level: _skip } }
      }
      {
        \__enumext_stop_list:
        \__enumext_stop_store_level:
      }
  }
```

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

`\__enumext_before_list:`

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

```
\cs_new_protected:Nn \__enumext_before_list:
  {
    \__enumext_vspace_above:
    \__enumext_before_args_exec:
    \__enumext_check_ans_active:
```

When the `mini-env` key is active it will set the value of the `\l__enumext_minipage_right_X_dim` to be the *width* of the `__enumext_mini_page` environment on the *"right side"*, using this value together with the value of the `\l__enumext_minipage_hsep_X_dim` set by the `mini-sep` key, the value of `\l__enumext_minipage_left_X_dim` will be set, which will be the *width* of `__enumext_mini_page` environment on the *"left side"*, always having a current `\linewidth` as *maximum width* between them.

```
    \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_minipage_add_space: is called and the __enumext_mini_page environment on the *"left side"* will be initialized followed by the *"vertical spacing"* applied to preserve the *"baseline"* between the *left* and *right* side environments. After these actions, the function __enumext_multicols_start: is called to handle the multicols environment.

```
        \bool_set_true:c { l__enumext_minipage_active_ \__enumext_level: _bool }
        \int_gincr:N \g__enumext_minipage_stat_int
        \__enumext_minipage_add_space:
        \noindent
        \__enumext_mini_page{ \dim_use:c { l__enumext_minipage_left_ \__enumext_level: _dim } }
      }
    \__enumext_multicols_start:
  }
```

(*End of definition for* __enumext_before_list:.)

__enumext_second_part: The function __enumext_second_part: first check the state of the boolean variable \l__enumext_minipage_active_X_bool, if it is "true" a small test will be executed to check if we have omitted the use of \miniright (the __enumext_mini_page environment has not been closed), then close __enumext_mini_page and add the *adjusted vertical space* \l__enumext_minipage_after_skip, otherwise we will close the multicols environment.

```
\cs_new_protected:Nn \__enumext_second_part:
  {
    \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
        \__enumext_unskip_unkern: % remove topsep + [partopsep]
        \end__enumext_mini_page
      }
      {
        \__enumext_multicols_stop:
      }
```

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

```
    \__enumext_after_stop_list:
    \__enumext_check_ans_key_hook:
    \__enumext_vspace_below:
    \bool_set_false:N \l__enumext_standar_bool
    \bool_if:NF \l__enumext_print_keyans_cmd_bool
      {
        \__enumext_standar_save_counter:
      }
  }
```

(*End of definition for* __enumext_second_part:.)

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

```
\cs_new_protected:Nn \__enumext_set_item_width:
  {
    \dim_set:Nn \itemwidth { \linewidth }
    \dim_compare:nT
```

```
      {
        \dim_use:c { l__enumext_listoffset_ \__enumext_level: _dim } != \c_zero_dim
      }
      {
        \dim_sub:Nn \itemwidth
          {
            \dim_use:c { l__enumext_listoffset_ \__enumext_level: _dim }
          }
      }
  }
```

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

`\__enumext_start_counter:` For compatibility with *tagged* PDF and since we are using legacy code for the implementation, we must set the initial value of the counters after the second argument to the list environment and before the first execution of `\item` , i.e. `\begin{list}{`⟨*arg one*⟩`}{`⟨*arg two*⟩`}\setcounter{enumX}`.

This is described in processing order of legacysetupcode in the block templates and we will apply the workaround provided by Frank Mittelbach.

```
\cs_new_protected:Nn \__enumext_start_counter:
  {
    \setcounter { enumX \__enumext_level: }
      {
        \int_eval:n { \int_use:c { l__enumext_start_ \__enumext_level: _int } - 1 }
      }
  }
```

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

`enumext` Now 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_start_counter:
    \__enumext_set_item_width:
    \__enumext_after_args_exec:
  }
  {
    \__enumext_second_part:
  }
```

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

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:
  }
```

13.43 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"*.

`\__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_name_and_start:
  % 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:` `\__enumext_keyans_multicols_start:` `\__enumext_keyans_multicols_stop:` `\__enumext_second_part_v:` Same implementation as the one used in the `enumext` environment.

```
\cs_new_protected:Nn \__enumext_before_list_v:
  {
    \__enumext_vspace_above_v:
    \__enumext_before_args_exec_v:
    \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
          }
        \bool_set_true:N \l__enumext_minipage_active_v_bool
        \int_gincr:N \g__enumext_minipage_stat_int
        \__enumext_keyans_minipage_add_space:
        \__enumext_mini_page{ \l__enumext_minipage_left_v_dim }
      }
    \__enumext_keyans_multicols_start:
  }
\cs_new_protected:Nn \__enumext_keyans_multicols_start:
  {
    \int_compare:nNnT { \l__enumext_columns_v_int } > { 1 }
      {
        \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
        \dim_zero:N \columnseprule % no rule here
        \bool_if:NF \l__enumext_minipage_active_v_bool
          {
            \skip_zero:N \multicolsep
            \__enumext_keyans_multi_addvspace:
          }
        \raggedcolumns
        \begin{multicols}{ \l__enumext_columns_v_int }
      }
  }
\cs_new_protected:Nn \__enumext_keyans_multicols_stop:
```

```
  {
    \int_compare:nNnTF { \l__enumext_columns_v_int } > { 1 }
      {
        \__enumext_stop_list:
        \end{multicols}
        \__enumext_unskip_unkern:
        \__enumext_unskip_unkern:
        \par\addvspace{ \l__enumext_multicols_below_v_skip }
      }
      {
        \__enumext_stop_list:
      }
  }
\cs_new_protected:Nn \__enumext_second_part_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
        \__enumext_unskip_unkern: % remove \topsep + [\partopsep]
        \end__enumext_mini_page
        \par\addvspace{ \l__enumext_minipage_after_skip }
      }
      {
        \__enumext_keyans_multicols_stop:
      }
    \bool_set_false:N \l__enumext_keyans_env_bool
    \__enumext_after_stop_list_v:
    \__enumext_vspace_below_v:
  }
```

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

`\__enumext_keyans_set_item_width:`

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

```
\cs_new_protected:Nn \__enumext_keyans_set_item_width:
  {
    \dim_set:Nn \itemwidth { \linewidth }
    \dim_compare:nT
      {
        \l__enumext_listoffset_v_dim != \c_zero_dim
      }
      {
        \dim_sub:Nn \itemwidth { \l__enumext_listoffset_v_dim }
      }
  }
```

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

`\__enumext_keyans_start_counter:`

For compatibility with *tagged* PDF and since we are using legacy code for the implementation, we must set the initial value of the counters after the second argument to the list environment and before the first execution of `\item` , i.e. `\begin{list}{`⟨*arg one*⟩`}{`⟨*arg two*⟩`}\setcounter{enumX}`.

```
\cs_new_protected:Nn \__enumext_keyans_start_counter:
  {
    \setcounter { enumXv } { \int_eval:n { \int_use:c { l__enumext_start_v_int } - 1 } }
  }
```

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

`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_keyans_start_counter:
    \__enumext_keyans_set_item_width:
    \__enumext_after_args_exec_v:
  }
  {
    \__enumext_check_starred_cmd:n { item }
    \__enumext_second_part_v:
  }
```

(*End of definition for* keyans. *This function is documented on page 16.*)

13.44 Tagging PDF support for non-standart list environments

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

To maintain a correct `list` structure when `\DocumentMetadata` is active, it is necessary to do some things manually using `tagpdf`[18] and `ltsockets`[20]. This implementation is an adaptation of my answer thanks to Ulrike Fischer's comments in How can I modify my `\item` redefinition to be compatible with `tagging-pdf`.

13.44.1 Socket for tagging support in enumext* and keyans*

start-list-tags
stop-start-tags
stop-list-tags
__enumext_start_list_tag:n
__enumext_stop_start_list_tag:
__enumext_stop_list_tag:n

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

```
\socket_new:nn {tagsupport/__enumext/starred}{ 1 }
\socket_new_plug:nnn {tagsupport/__enumext/starred} {start-list-tags}
  {
    \tag_resume:n {#1}
    \tag_mc_end_push:
      \tag_struct_begin:n {tag=LI}
        \tag_struct_begin:n {tag=Lbl}
           \tag_mc_begin:n {tag=Lbl}
  }
\socket_new_plug:nnn {tagsupport/__enumext/starred} {stop-start-tags}
  {
           \tag_mc_end:
        \tag_struct_end:n {tag=Lbl}
        \tag_struct_begin:n {tag=LBody}
          \tag_struct_begin:n {tag=text-unit}
            \tag_struct_begin:n {tag=text}
  }
\socket_new_plug:nnn {tagsupport/__enumext/starred} {stop-list-tags}
  {
            \tag_struct_end:n {tag=text}
          \tag_struct_end:n {tag=text-unit}
        \tag_struct_end:n {tag=LBody}
      \tag_struct_end:n {tag=LI}
    \tag_mc_begin_pop:n {}
    \tag_suspend:n {#1}
  }
```

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

```
\cs_new_protected_nopar:Npn \__enumext_start_list_tag:n #1
  {
    \IfDocumentMetadataT
      {
        \socket_assign_plug:nn {tagsupport/__enumext/starred} {start-list-tags}
        \socket_use:nn {tagsupport/__enumext/starred} {#1}
      }
   }
\cs_new_protected_nopar:Nn \__enumext_stop_start_list_tag:
  {
    \IfDocumentMetadataT
      {
        \socket_assign_plug:nn {tagsupport/__enumext/starred} {stop-start-tags}
        \socket_use:nn {tagsupport/__enumext/starred} { }
      }
   }
```

```
\cs_new_protected_nopar:Npn \__enumext_stop_list_tag:n #1
  {
    \IfDocumentMetadataT
      {
        \socket_assign_plug:nn {tagsupport/__enumext/starred} {stop-list-tags}
        \socket_use:nn {tagsupport/__enumext/starred} {#1}
      }
   }
```

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

13.44.2 Socket for tagging support in keyanspic

start-list-tags
stop-start-tags
stop-list-tags
__enumext_anspic_start_list_tag:
__enumext_anspic_stop_start_list_tag:
__enumext_anspic_stop_list_tag:

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

```
\socket_new:nn {tagsupport/__enumext/keyanspic}{ 0 }
\socket_new_plug:nnn {tagsupport/__enumext/keyanspic} {start-list-tags}
  {
    \tag_resume:n {keyanspic}
    \tag_mc_end_push:
      \tag_struct_begin:n {tag=LI}
        \tag_struct_begin:n {tag=Lbl}
          \tag_mc_begin:n {tag=Lbl}
  }
\socket_new_plug:nnn {tagsupport/__enumext/keyanspic} {stop-start-tags}
  {
          \tag_mc_end:
        \tag_struct_end:n {tag=Lbl}
        \tag_struct_begin:n {tag=LBody}
          \tag_struct_begin:n {tag=text-unit}
            \tag_struct_begin:n {tag=text}
              \tag_mc_begin:n {tag=text}
  }
\socket_new_plug:nnn {tagsupport/__enumext/keyanspic} {stop-list-tags}
  {
              \tag_mc_end:
            \tag_struct_end:n {tag=text}
          \tag_struct_end:n {tag=text-unit}
        \tag_struct_end:n {tag=LBody}
      \tag_struct_end:n {tag=LI}
    \tag_mc_begin_pop:n {}
    \tag_suspend:n {keyanspic}
  }
```

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

```
\cs_new_protected_nopar:Nn \__enumext_anspic_start_list_tag:
  {
    \IfDocumentMetadataT
      {
        \socket_assign_plug:nn {tagsupport/__enumext/keyanspic} {start-list-tags}
        \socket_use:n {tagsupport/__enumext/keyanspic}
      }
   }
\cs_new_protected_nopar:Nn \__enumext_anspic_stop_start_list_tag:
  {
    \IfDocumentMetadataT
      {
        \socket_assign_plug:nn {tagsupport/__enumext/keyanspic} {stop-start-tags}
        \socket_use:n {tagsupport/__enumext/keyanspic}
      }
   }
\cs_new_protected_nopar:Nn \__enumext_anspic_stop_list_tag:
  {
    \IfDocumentMetadataT
      {
        \socket_assign_plug:nn {tagsupport/__enumext/keyanspic} {stop-list-tags}
        \socket_use:n {tagsupport/__enumext/keyanspic}
      }
   }
```

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

13.45 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 as replacement for \item command and placed inside minipage environments, with the ⟨*label*⟩ centered *"above"* or *"below"*, adjusting *widths* and *position* according to the options passed to the environment.

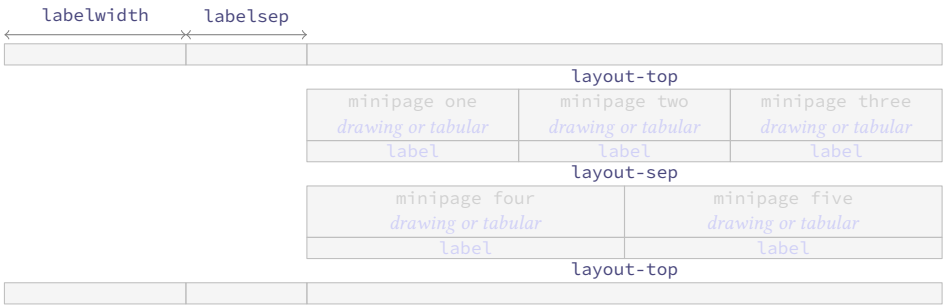


Figure 12: Representation of the keyanspic spacing in enumext.

In order for the keyanspic environment and the \anspic command to work correctly, we need to set and export some variables in the first part of the environment definition and pass them to \anspic which is executed in the second part of the environment. This implementation is adapted from the answer given by Enrico Gregorio (@egreg) in How to process the body of an environment and divide it by a \macro?.

13.45.1 The environment keyanspic

label-pos
label-sep
layout-sty
layout-sep
layout-top
mark-ans
mark-pos
mark-sep
save-sep
wrap-opt
wrap-ans*
show-ans
show-pos

First we define the key that allows us to process the position of the ⟨*label*⟩ centered "above" or "below" which will be label-pos, the vertical separation of these from *drawing or tabular* will be handled with the key label-sep. The *"layout style"* will be handled with the key layout-sty will take two values separated by comma {⟨*n° upper, n° lower*⟩} and will determine the number of minipage environments in which all arguments of \anspic will be printed at the "upper" and "lower" within the environments separated by the value of the key layout-sep. The vertical space "top" and "bottom" of the environment will be handled with the key layout-top.

```
\keys_define:nn { enumext / keyanspic }
  {
    label-pos .choice:,
    label-pos / above   .code:n =
                              \bool_set_true:N \l__enumext_anspic_label_above_bool
                              \str_set:Nn \l__enumext_anspic_mini_pos_str { t },
    label-pos / below   .code:n =
                              \bool_set_false:N \l__enumext_anspic_label_above_bool
                              \str_set:Nn \l__enumext_anspic_mini_pos_str { b },
    label-pos / unknown .code:n =
                              \msg_error:nneee { enumext } { unknown-choice }
                                { label-pos } { above,~ below } { \exp_not:n {#1} },
    label-pos  .initial:n        = below,
    label-pos  .value_required:n = true,
    label-sep  .skip_set:N       = \l__enumext_anspic_label_sep_skip,
    label-sep  .value_required:n = true,
    layout-sty .tl_set:N         = \l__enumext_anspic_layout_style_tl,
    layout-sty .value_required:n = true,
    layout-sep .code:n           = \keys_set:nn { enumext / keyans } { parsep = #1 },
    layout-sep .value_required:n = true,
    layout-top .code:n           = \keys_set:nn { enumext / keyans } { topsep = #1 },
    layout-top .value_required:n = true,
    mark-ans   .code:n           = \keys_set:nn { enumext / keyans } { mark-ans = #1 },
    mark-ans   .value_required:n = true,
    mark-pos   .code:n           = \keys_set:nn { enumext / keyans } { mark-pos = #1 },
    mark-pos   .value_required:n = true,
    mark-sep   .code:n           = \keys_set:nn { enumext / keyans } { mark-sep = #1 },
    mark-sep   .value_required:n = true,
    save-sep   .code:n           = \keys_set:nn { enumext / keyans } { save-sep = #1 },
    save-sep   .value_required:n = true,
    wrap-opt   .code:n           = \keys_set:nn { enumext / keyans } { wrap-opt = #1 },
    wrap-opt   .value_required:n = true,
    wrap-ans*  .code:n           = \keys_set:nn { enumext / keyans } { wrap-ans* = #1 },
    wrap-ans*  .value_required:n = true,
    show-ans   .code:n           = \keys_set:nn { enumext / keyans } { show-ans = #1 },
    show-ans   .value_required:n = true,
    show-pos   .code:n           = \keys_set:nn { enumext / keyans } { show-pos = #1 },
    show-pos   .value_required:n = true,
```

```
    unknown     .code:n           = {
                                      \tl_set:Nn \l__enumext_envir_name_tl { keyanspic }
                                      \__enumext_keyans_unknown_keys:n {#1}
                                    },
  }
```

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

`\__enumext_keyans_pic_safe_exec:`
`\__enumext_keyans_pic_parse_keys:n`
`\__enumext_keyans_pic_skip_abs:N`
`\__enumext_keyans_pic_arg_two:`

The function `\__enumext_keyans_pic_safe_exec:` check the nested 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_name_and_start:
  }
```

Parse `[`⟨*key* = *val*⟩`]` for `keyanspic` environment.

```
\cs_new_protected:Npn \__enumext_keyans_pic_parse_keys:n #1
  {
    \tl_if_novalue:nF {#1}
      {
        \keys_set:nn { enumext / keyanspic } {#1}
      }
  }
```

The function `\__enumext_keyans_pic_skip_abs:N` will return a positive value `\parsep` from `keyans` environment.

```
\cs_new_protected:Npn \__enumext_keyans_pic_skip_abs:N #1
  {
    \dim_compare:nNnT { #1 } < { \c_zero_dim }
      {
        \skip_set:Nn #1 { -#1 }
      }
  }
```

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

```
\cs_new_protected:Npn \__enumext_keyans_pic_arg_two:
  {
    \bool_set_false:N \l__enumext_leftmargin_tmp_v_bool
    \__enumext_list_arg_two_v:
    \__enumext_keyans_pic_skip_abs:N \parsep
```

Now we increment the counter `enumXv` of the `keyans` environment and save the *total height* of the ⟨*label*⟩ in `\l__enumext_anspic_label_htdp_dim` used by `\anspic` and we will adjust the values of `\parsep` only if the key `label-pos` is set to *below*.

```
    \bool_if:NF \l__enumext_anspic_label_above_bool
      {
        \stepcounter { enumXv }
        \hbox_set:Nn \l__enumext_anspic_label_box { \l__enumext_label_v_tl }
        \dim_set:Nn \l__enumext_anspic_label_htdp_dim
          {
            \box_ht_plus_dp:N \l__enumext_anspic_label_box
          }
        \skip_add:Nn \parsep
          {
            \l__enumext_anspic_label_htdp_dim
            + \box_dp:N \strutbox
            + \l__enumext_anspic_label_sep_skip
          }
      }
```

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

```
    \dim_add:Nn \leftmargin { -\l__enumext_labelwidth_v_dim - \l__enumext_labelsep_v_dim }
    \ignorespaces
    \skip_add:Nn \topsep { 0.5\box_dp:N \strutbox }
    \dim_zero:N  \listparindent
    \skip_zero:N \partopsep
    \skip_zero:N \itemsep
  }
```

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

keyanspic

Now we define the environment keyanspic. For compatibility with *tagged* PDF we must use the \begin{list} form and a lot of conditional code using \IfDocumentMetadataTF. We will first stop the code for automatic *tagged* PDF for list environments, redefine \item so that it cannot be used, and stop the code for automatic *tagged* PDF for the keyanspic environment.

```
\NewDocumentEnvironment{keyanspic}{ o }
  {
    \__enumext_keyans_pic_safe_exec:
    \__enumext_keyans_pic_parse_keys:n {#1}
    \begin{list} { } { \__enumext_keyans_pic_arg_two: }
    \IfDocumentMetadataT
      {
        \tag_suspend:n {list}
      }
    \item[] \scan_stop:
    \RenewDocumentCommand \item {}
      {
        \msg_error:nn { enumext } { keyanspic-item-cmd }
      }
    \IfDocumentMetadataT
      {
        \tag_resume:n {keyanspic}
        \tag_tool:n {para/tagging=false}
        \tag_suspend:n {keyanspic}
      }
  }
```

In the second part of the environment definition we will manually place our code for *tagged* PDF and execute the command \anspic using the `\__enumext_anspic_exec:` function.

```
  {
    \IfDocumentMetadataT
      {
        \tag_resume:n {keyanspic}
        \tag_mc_end_push:
        \tag_struct_begin:n {tag=L,attribute=enumerate}
      }
    \__enumext_anspic_exec:
    \IfDocumentMetadataT
      {
        \tag_suspend:n {keyanspic}
      }
    \end{list}
    \IfDocumentMetadataT
      {
        \tag_struct_end:n {tag=L}
        \tag_mc_begin_pop:n {}
        \tag_struct_end:n {tag=L}
        \tag_mc_begin_pop:n {}
      }
```

Finally we check if \anspic* has been used, set the counter enumXvi to zero and apply our "adjusted" vertical space bottom.

```
    \__enumext_check_starred_cmd:n { anspic }
    \setcounter { enumXvi } { 0 }
    \bool_if:NTF \l__enumext_anspic_label_above_bool
      {
        \par\addvspace{ 0.5\box_dp:N \strutbox }
      }
      {
        \par
```

```
        \addvspace
          {
            \dim_eval:n
              {
                \l__enumext_anspic_label_htdp_dim + \box_ht_plus_dp:N \strutbox
                + \l__enumext_anspic_label_sep_skip + \l__enumext_topsep_v_skip
              }
          }
      }
  }
```

(*End of definition for* keyanspic. *This function is documented on page 17.*)

13.45.2 The command \anspic

The \anspic command take three arguments, the *starred versions* \anspic*[⟨*content*⟩] *store* the current ⟨*label*⟩ next to the *optional argument* [⟨*content*⟩] in the *sequence* and *prop list* defined by save-ans key. The third *mandatory argument* {⟨*drawing or tabular*⟩} is NOT *stored* in the *sequence* or *prop list.*

One of the complications here to make the keyanspic environment compatible with *tagged* PDF is the position of ⟨*label*⟩, the \anspic command processes the arguments in order, where #1 and #2 correspond to ⟨*label*⟩ and #3 to the mandatory argument and puts all this inside a minipage environment. If #1 and #2, that is ⟨*label*⟩, is above #3 there are no problems with *tagged* PDF, but if #3 comes first the list created with *tagged* PDF will not be correct.

\anspic
__enumext_anspic_body_dim:n
__enumext_anspic_label:nn
__enumext_anspic_label_pos:nnn
__enumext_anspic_args:nnn
__enumext_anspic_print:n
__enumext_anspic_print:e
__enumext_anspic_print:V
__enumext_anspic_row:n
__enumext_anspic_exec:

We check that the command is active in the keyanspic environment only if the save-ans key is present, otherwise we return an error. The three arguments are handled by the function __enumext_anspic_args:nnn and stored in the sequence \l__enumext_anspic_args_seq which is processed by the keyanspic environment.

```
\NewDocumentCommand \anspic { s o +m }
  {
    \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 }
      }
    \seq_put_right:Nn \l__enumext_anspic_args_seq
      {
        \__enumext_anspic_args:nnn { #1 } { #2 } { #3 }
      }
  }
```

The __enumext_anspic_body_dim:n function will set the value of \l__enumext_anspic_body_htdp_dim equal to the "height plus depth" of the *mandatory argument* if the key label-pos is set "below".

```
\cs_new_protected:Npn \__enumext_anspic_body_dim:n #1
  {
    \bool_if:NF \l__enumext_anspic_label_above_bool
      {
        \IfDocumentMetadataT
          {
            \tag_suspend:n {keyanspic}
          }
        \vbox_set:Nn \l__enumext_anspic_body_box { #1 }
        \dim_set:Nn \l__enumext_anspic_body_htdp_dim
          {
            \box_ht_plus_dp:N \l__enumext_anspic_body_box
          }
        \IfDocumentMetadataT
          {
            \tag_resume:n {keyanspic}
          }
      }
  }
```

The __enumext_anspic_label:nn function will process inside \makebox the *starred argument* '*' and *optional argument* passed to the command. Here we will store the ⟨*label*⟩ and *optional argument* in *prop list* and *sequence* and execute the show-ans, show-pos, font, wrap-label, wrap-ans* and wrap-opt keys.

```
\cs_new_protected:Npn \__enumext_anspic_label:nn #1 #2
  {
    \makebox[ \l__enumext_anspic_mini_width_dim ][ c ]
      {
        \bool_if:nTF { #1 }
          {
            \bool_set_true:N \l__enumext_item_wrap_key_bool
            \bool_set_true:N \l__enumext_wrap_label_v_bool
            \__enumext_keyans_save_item_opt:n { #2 }
            \__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
            \__enumext_keyans_show_ans:
            \__enumext_keyans_show_pos:
            \makebox[ \l__enumext_labelwidth_v_dim ][c]
              {
                \tl_use:N \l__enumext_label_font_style_v_tl
                \__enumext_keyans_wrapper_label:n { \l__enumext_label_vi_tl }
              }
            \skip_horizontal:n { \l__enumext_labelsep_v_dim }
            \__enumext_keyans_show_item_opt:
          }
          {
            \bool_set_false:N \l__enumext_item_wrap_key_bool
            \tl_use:N \l__enumext_label_font_style_v_tl
            \__enumext_wrapper_label_v:n { \l__enumext_label_vi_tl }
          }
      }
  }
```

The function `\__enumext_anspic_label_pos:nnn` will be in charge of handling the *"counter"* and the position of the ⟨*label*⟩, set by `label-pos` key which will have the same configuration as the `keyans` environment.

```
\cs_new_protected:Npn \__enumext_anspic_label_pos:nnn #1 #2 #3
  {
    \stepcounter { enumXvi }
    \__enumext_anspic_body_dim:n { #3 }
    \bool_if:NTF \l__enumext_anspic_label_above_bool
      {
        \__enumext_anspic_label:nn { #1 } { #2 }
      }
      {
        \raisebox
          {
            -\dim_eval:n
              {
                \l__enumext_anspic_label_htdp_dim
                + \l__enumext_anspic_body_htdp_dim
                + \box_dp:N \strutbox
                + \l__enumext_anspic_label_sep_skip
              }
          }
          [ 0pt ] [ 0pt ]
          {
            \__enumext_anspic_label:nn { #1 } { #2 }
          }
      }
  }
%
```

The `\__enumext_anspic_args:nnn` function will be responsible for placing the code compatible with *tagged* PDF and the arguments within the `\l__enumext_anspic_args_seq` sequence which will be processed by the `\__enumext_anspic_print:n` function in the second part of the definition of the `keyanspic` environment.

```
\cs_new_protected:Nn \__enumext_anspic_args:nnn
  {
    \__enumext_anspic_start_list_tag:
    \__enumext_anspic_label_pos:nnn { #1 } { #2 } { #3 }
    \__enumext_anspic_stop_start_list_tag:
    \bool_if:NTF \l__enumext_anspic_label_above_bool
      {
        \\[\l__enumext_anspic_label_sep_skip] #3
      }
```

```
      {
        \\ #3
      }
    \__enumext_anspic_stop_list_tag:
  }
```

The value {⟨*n° upper, n° lower*⟩} passed to the `layout-sty` key is split by comma and is handled directly by the function `\__enumext_anspic_print:n` and passed to the function `\__enumext_anspic_row:n`.

```
\cs_new_protected:Nn \__enumext_anspic_print:n
  {
    \clist_map_function:nN { #1 } \__enumext_anspic_row:n
  }
\cs_generate_variant:Nn \__enumext_anspic_print:n { e, V }
```

The function `\__enumext_anspic_row:n` will set the *widths* for the `minipage` environments and place *all arguments* passed to `\anspic` *saved* in the `\l__enumext_anspic_args_seq` sequence inside them.

```
\cs_new_protected:Nn \__enumext_anspic_row:n
  {
    \dim_set:Nn \l__enumext_anspic_mini_width_dim { \linewidth / #1 }
    \int_set:Nn \l__enumext_anspic_above_int { \l__enumext_anspic_below_int }
    \int_set:Nn \l__enumext_anspic_below_int { \l__enumext_anspic_above_int + #1 }
    \int_step_inline:nnn
      { \l__enumext_anspic_above_int + 1 }
      { \l__enumext_anspic_below_int }
      {
        \IfDocumentMetadataT
          {
            \tag_suspend:n {minipage}
          }
        \begin{minipage}[ \l__enumext_anspic_mini_pos_str ]{ \l__enumext_anspic_mini_width_dim }
          \centering
          \seq_item:Nn \l__enumext_anspic_args_seq { ##1 }
        \end{minipage}
        \IfDocumentMetadataT
          {
            \tag_resume:n {minipage}
          }
      }
    \par
  }
```

The `\__enumext_anspic_exec:` function will execute all the code in the `\anspic` command in the second argument of the `keyanspic` environment definition. If the key `layout-sty` is not set, everything will be printed on a *single line*.

```
\cs_new_protected:Nn \__enumext_anspic_exec:
  {
    \tl_if_empty:NTF \l__enumext_anspic_layout_style_tl
      {
        \__enumext_anspic_print:e { \seq_count:N \l__enumext_anspic_args_seq }
      }
      {
        \__enumext_anspic_print:V \l__enumext_anspic_layout_style_tl
      }
  }
```

(*End of definition for* `\anspic` *and others. This function is documented on page 18.*)

13.46 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` using `\RenewDocumentCommand` as in the vertical *non starred* versions.

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

A side effect is the limitation of using `\item` in this way *without* using `\RenewDocumentCommand`, which loses the original definition and affects the *standard list environments* provided by LaTeX and any environment defined using base `list` environment, including: `itemize`, `enumerate`, `description`, `quote`, `quotation`, `verse`,

center, flushleft, flushright, verbatim, tabbing, trivlist, list and all environments created with \newtheorem.

One way to get around this is to use something like:

```
\AddToHook{env/enumerate/before}{recover original \item definition}
```

inside minipage, but in my partial tests this does not have the desired effect and the vertical and horizontal spacing is distorted. For now this will remain as a limitation and I will see if it is feasible to implement it in the future.

For compatibility with the *tagged* PDF we close the environments according to the presence or not of the mini-env key.

13.46.1 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:`.)

13.46.2 Functions for join item columns

`\__enumext_starred_joined_item_vii:n`
`\__enumext_starred_joined_item_viii:n`

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

```
\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 \itemwidth \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 \itemwidth \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 }
```

```
4768            { \int_use:N \l__enumext_columns_viii_int }
4769          \int_set:Nn \l__enumext_joined_item_viii_int
4770            {
4771              \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1
4772            }
4773        }
4774      \int_compare:nNnT
4775        { \l__enumext_joined_item_viii_int }
4776          >
4777        { \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1 }
4778        {
4779          \msg_warning:nnee { enumext } { item-joined-columns }
4780            { \int_use:N \l__enumext_joined_item_viii_int }
4781            {
4782              \int_eval:n
4783                { \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1 }
4784            }
4785          \int_set:Nn \l__enumext_joined_item_viii_int
4786            {
4787              \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1
4788            }
4789        }
4790      \int_compare:nNnTF { \l__enumext_joined_item_viii_int } > { 1 }
4791        {
4792          \int_set_eq:NN \l__enumext_joined_item_aux_viii_int \l__enumext_joined_item_viii_int
4793          \int_decr:N \l__enumext_joined_item_aux_viii_int
4794          \int_add:Nn \l__enumext_item_column_pos_viii_int { \l__enumext_joined_item_aux_viii_int }
4795          \int_gadd:Nn \g__enumext_item_count_all_viii_int { \l__enumext_joined_item_aux_viii_int }
4796          \dim_set:Nn \l__enumext_joined_width_viii_dim
4797            {
4798              \l__enumext_item_width_viii_dim * \l__enumext_joined_item_viii_int
4799              + ( \l__enumext_labelwidth_viii_dim + \l__enumext_labelsep_viii_dim
4800                  + \l__enumext_columns_sep_viii_dim
4801                )*\l__enumext_joined_item_aux_viii_int
4802            }
4803          \dim_set_eq:NN \itemwidth \l__enumext_joined_width_viii_dim
4804        }
4805        {
4806          \dim_set_eq:NN \l__enumext_joined_width_viii_dim \l__enumext_item_width_viii_dim
4807          \dim_set_eq:NN \itemwidth \l__enumext_item_width_viii_dim
4808        }
4809    }
```

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

13.46.3 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_page` environment on the *"right side"* is executed *"after"* closing the environment, so it is necessary to make a global copy of the variable `\l__enumext_minipage_right_vii_dim` in the variable `\g__enumext_minipage_right_vii_dim`.

```
4810  \cs_new_protected:Nn \__enumext_start_mini_vii:
4811    {
4812      \dim_compare:nNnT { \l__enumext_minipage_right_vii_dim } > { \c_zero_dim }
4813        {
4814          \dim_set:Nn \l__enumext_minipage_left_vii_dim
4815            {
4816              \linewidth
4817              - \l__enumext_minipage_right_vii_dim
4818              - \l__enumext_minipage_hsep_vii_dim
4819            }
4820          \bool_set_true:N \l__enumext_minipage_active_vii_bool
4821          \dim_gset_eq:NN
4822            \g__enumext_minipage_right_vii_dim
4823            \l__enumext_minipage_right_vii_dim
4824          \__enumext_mini_addvspace_vii:
4825          \nointerlineskip\noindent
4826          \__enumext_mini_page{ \l__enumext_minipage_left_vii_dim }
4827        }
4828    }
```

The function `\__enumext_stop_mini_vii:` closes the `__enumext_mini_page` environment on the *"left side"*, applies `\hfill` and set the variable `\g__enumext_minipage_active_vii_bool` to *"true"* which will

be used in the function `\__enumext_after_env:nn` to execute the `minipage` on the *"right side"*. At this point we will execute the `\__enumext_stop_list:` and `\__enumext_stop_store_level_vii:` functions stopping the `list` environment and the level saving mechanism for storage in *sequence* of the `\anskey` command and `anskey*` environment. This function is passed to the `\__enumext_after_list_vii:` function in the second part of the `enumext*` environment definition (§13.47).

```
\cs_new_protected:Nn \__enumext_stop_mini_vii:
  {
    \bool_if:NTF \l__enumext_minipage_active_vii_bool
      {
        \__enumext_stop_list:
        \__enumext_stop_store_level_vii:
        \IfDocumentMetadataT { \tag_resume:n {enumext*} }
        \end__enumext_mini_page
        \hfill
        \bool_gset_true:N \g__enumext_minipage_active_vii_bool
      }
      {
        \__enumext_stop_list:
        \__enumext_stop_store_level_vii:
      }
   }
```

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

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

```
\__enumext_after_env:nn {enumext*}
  {
    \bool_if:NT \g__enumext_minipage_active_vii_bool
      {
        \__enumext_minipage:w [ t ] { \g__enumext_minipage_right_vii_dim }
          \legacy_if_gset_false:n { @minipage }
          \skip_vertical:N \c_zero_skip
          \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
          \skip_vertical:N \c_zero_skip
        \__enumext_endminipage:
        \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
  }
```

`\__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
      \__enumext_mini_page{ \l__enumext_minipage_left_viii_dim }
    }
  }
\cs_new_protected:Nn \__enumext_stop_mini_viii:
  {
    \bool_if:NTF \l__enumext_minipage_active_viii_bool
      {
        \__enumext_stop_list:
        \IfDocumentMetadataT { \tag_resume:n {keyans*} }
        \end__enumext_mini_page
        \hfill
        \bool_gset_true:N \g__enumext_minipage_active_viii_bool
      }
      {
        \__enumext_stop_list:
      }
  }
\__enumext_after_env:nn {keyans*}
  {
    \bool_if:NT \g__enumext_minipage_active_viii_bool
      {
        \__enumext_mini_page{ \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_page
        \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:`.)

13.47 The environment `enumext*`

enumext*

First we will generate the environment and we will give a temporary definition to `\__enumext_stop_item_tmp_vii:` equal to `\__enumext_first_item_tmp_vii:` and next to `\item` equal to `\__enumext_start_item_tmp_vii:` which we will redefine later. Unlike the implementation used by the shortlst package, we will not set the values of `\rightskip` and `\@rightskip` equal to `\@flushglue` whose value is `0.0pt plus 1.0 fil`, in the tests I have performed this fails in some circumstances and different results are obtained when using pdfTEX and LuaTEX.

```
\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:
      }
```

```
    \setcounter { enumXvii } { \int_eval:n { \int_use:c { l__enumext_start_vii_int } - 1 } }
    \IfDocumentMetadataT { \tag_suspend:n {enumext*} }
    \__enumext_starred_columns_set_vii:
    \item[] \scan_stop:
    \cs_set_eq:NN \__enumext_stop_item_tmp_vii: \__enumext_first_item_tmp_vii:
    \cs_set_eq:NN \item \__enumext_start_item_tmp_vii:
    \ignorespaces
  }
  {
    \IfDocumentMetadataT { \tag_struct_end:n {tag=text-unit} }
    \__enumext_stop_item_tmp_vii:
    \__enumext_remove_extra_parsep_vii:
    \__enumext_after_list_vii:
  }
```

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

`\__enumext_safe_exec_vii:`

We will first call the function `\__enumext_is_not_nested:` which sets `\g__enumext_starred_bool` to true if we are NOT nested within `enumext`, then call the function `\__enumext_internal_mini_page:` to create the environment `__enumext_mini_page`, 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_is_not_nested:
    \__enumext_internal_mini_page:
    \int_incr:N \l__enumext_level_h_int
    \int_compare:nNnT { \l__enumext_level_h_int } > { 1 }
      {
        \msg_error:nn { enumext } { nested }
      }
    \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
      {
        \msg_error:nnn { enumext } { nested-horizontal } { keyans*}
      }
    \bool_set_true:N \l__enumext_starred_bool
    \bool_set_false:N \l__enumext_standar_bool
    \__enumext_is_on_first_level:
  }
```

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

`\__enumext_parse_keys_vii:n`

We will first check the state of the variable `\l__enumext_resume_count_vii_bool` set by the key `resume` and call the function `\__enumext_resume_last_counter:` if it is "true", then we will clear the variable `\l__enumext_series_name_str` used by the key `series`, process the environment `[`⟨*key* = *val*⟩`]` and execute the function `\__enumext_parse_series:n` 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.

Here it is necessary to check the status of `\l__enumext_resume_count_vii_bool` in case the key `resume` is set using `\setenumext{enumext*}{resume}`.

```
\cs_new_protected:Npn \__enumext_parse_keys_vii:n #1
  {
    \bool_if:NT \l__enumext_resume_count_vii_bool
      {
        \__enumext_resume_last_counter:
      }
    \tl_if_novalue:nF {#1}
      {
         \str_clear:N \l__enumext_series_name_str
         \keys_set:nn { enumext / enumext* } {#1}
         \bool_if:NF \l__enumext_print_keyans_cmd_bool
           {
             \__enumext_parse_series:n {#1}
           }
        \__enumext_store_active_keys_vii:n {#1}
      }
  }
```

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

`\__enumext_before_list_vii:` The function `\__enumext_before_list_vii:` first calls the function `\__enumext_vspace_above_vii:` used by the keys `above` and `above*`, then calls the function `\__enumext_check_ans_active:` for the check answer mechanism and finally calls the functions `\__enumext_before_args_exec:` and `\__enumext_start_mini_vii:` used by the keys `before*`, `mini-env`, `mini-right` and `mini-right*`.

```
\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:` which internally calls `\__enumext_stop_list:` and `\__enumext_stop_store_level_vii:` (§13.46.3) 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
    \bool_if:NF \l__enumext_print_keyans_cmd_bool
      {
        \__enumext_starred_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 *"storing structure"* mechanism in *sequence* for `\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:`.)

13.47.1 The command \item in enumext*

__enumext_first_item_tmp_vii:

The __enumext_first_item_tmp_vii: function will remove horizontal space equal to \labelwidth plus \labelsep to the left of the *"first"* \item in the environment at the point of execution of this function, where it is equal to the __enumext_stop_item_tmp_vii: function inside the environment body definition.

```
\cs_new_protected_nopar:Nn \__enumext_first_item_tmp_vii:
  {
    \skip_horizontal:n
      {
        -\l__enumext_labelwidth_vii_dim - \l__enumext_labelsep_vii_dim
      }
    \ignorespaces
  }
```

(*End of definition for* __enumext_first_item_tmp_vii:.)

__enumext_start_item_tmp_vii:
__enumext_item_peek_args_vii:
__enumext_joined_item_vii:w
__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

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:
  }
```

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) }
  }
```

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 }
  }
```

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 ] \ignorespaces
      }
  }
```

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 ] \ignorespaces
      }
  }
\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 ] \ignorespaces
  }
```

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

__enumext_fake_make_label_vii:n

The __enumext_fake_make_label_vii:n function will be in charge of handling our definition of \item. First we increment the counter enumXvii for the enumerated items and activate support for the *check answers* mechanism, followed by support for \item*[⟨*symbol*⟩][⟨*offset*⟩] if present, then the wrap-label and wrap-label* keys which we execute using \makebox whose width will be given by the labelwidth key and position by the align key, inside the argument of this we will execute the font key together with the function defined by the wrap-label or wrap-label* keys. Finally we execute the labelsep key applying a \skip_horizontal:N and \ignorespaces.

For compatibility with *tagged* PDF and hyperref when an environment enumext is nested in enumext* and the key save-ans is not active need setting the \if@hyper@item switch to *"true"*. The explanation for this is given by the master Heiko Oberdiek on \refstepcounter{enumi} twice (or more) creates destination with the same identifier. This patch is only needed if you are running pdflatex and not if you are running lualatex

```
\cs_new_protected_nopar:Npn \__enumext_fake_make_label_vii:n #1
  {
    \legacy_if:nT { @noitemarg }
      {
        \legacy_if_set_false:n { @noitemarg }
        \legacy_if:nT { @nmbrlist }
          {
            \IfDocumentMetadataT
              {
                \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
              }
          }
      }
    \bool_if:NT \l__enumext_item_starred_vii_bool
      {
        \tl_if_blank:VT \g__enumext_item_symbol_aux_vii_tl
```

```
5128            {
5129              \tl_gset_eq:NN
5130                \g__enumext_item_symbol_aux_vii_tl \l__enumext_item_symbol_vii_tl
5131            }
5132          \mode_leave_vertical:
5133          \skip_horizontal:n { -\l__enumext_item_symbol_sep_vii_dim }
5134          \hbox_overlap_left:n { \g__enumext_item_symbol_aux_vii_tl }
5135          \skip_horizontal:N \l__enumext_item_symbol_sep_vii_dim
5136          \tl_gclear:N \g__enumext_item_symbol_aux_vii_tl
5137        }
5138      \makebox[ \l__enumext_labelwidth_vii_dim ][ \l__enumext_align_label_vii_str ]
5139        {
5140          \tl_use:N \l__enumext_label_font_style_vii_tl
5141          \bool_if:NTF \l__enumext_wrap_label_vii_bool
5142            {
5143              \__enumext_wrapper_label_vii:n {#1}
5144            }
5145            { #1 }
5146        }
5147      \skip_horizontal:N \l__enumext_labelsep_vii_dim \ignorespaces
5148    }
```

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

13.47.2 Real definition of `\item` in enumext*

The functions `\__enumext_start_item_vii:w` and `\__enumext_stop_item_vii:` executing the true definition of `\item` inside the `enumext*` environment, unlike the implementation in `shortlst` we will NOT use an extra group and the plain form of the `lrbox` environment.

`\__enumext_start_item_vii:w`
`\__enumext_stop_item_vii:`

The first thing we will do is set the value of `\__enumext_stop_item_tmp_vii:` equal to `\__enumext_stop_item_vii:` which we will define later, after that we will start capturing `\item` and *"item content"* in a *horizontal box* where the width will be `\itemwidth` plus `\labelwidth` plus `\labelsep`.

```
5149 \cs_new_protected_nopar:Npn \__enumext_start_item_vii:w [#1]
5150   {
5151     \cs_set_eq:NN \__enumext_stop_item_tmp_vii: \__enumext_stop_item_vii:
5152     \hbox_set_to_wd:Nnw \l__enumext_item_text_vii_box
5153       {
5154         \l__enumext_joined_width_vii_dim
5155         + \l__enumext_labelwidth_vii_dim
5156         + \l__enumext_labelsep_vii_dim
5157       }
```

Redefine the `\footnote` command.

```
5158       \__enumext_renew_footnote_starred:
```

Now we insert our *sockets* for *tagging* PDF support and run `\item`.

```
5159       \__enumext_start_list_tag:n {enumext*}
5160       \__enumext_fake_make_label_vii:n {#1}
5161       \__enumext_stop_start_list_tag:
```

Finally we open the `minipage` environment, capture the *"item content"*, make `\parindent` take the value of the key `listparindent` and `\parskip` take the value of the key `parsep`, then execute the keys `itemindent` and `first`.

Here the use of `\unskip` and `\skip_horizontal:n` with the value of `listparindent` is necessary, otherwise an unwanted space is created when using `\item[⟨opt⟩]` and the value passed to the key `itemindent` is incremented.

```
5162       \__enumext_minipage:w [ t ]{ \l__enumext_joined_width_vii_dim }
5163         \dim_set_eq:NN \parindent \l__enumext_listparindent_vii_dim
5164         \skip_set_eq:NN \parskip \l__enumext_parsep_vii_skip
5165         \__enumext_unskip_unkern:
5166         \__enumext_unskip_unkern:
5167         \skip_horizontal:n { -\l__enumext_listparindent_vii_dim } \ignorespaces
5168         \tl_use:N \l__enumext_fake_item_indent_vii_tl
5169         \tl_use:N \l__enumext_after_list_args_vii_tl
5170   }
```

The `\__enumext_stop_item_vii:` function will finish the fetching `\item` and *"item content"* by closing the `minipage` environment, the *sockets* for *tagging* PDF and the *horizontal box*.

```
5171 \cs_new_protected_nopar:Nn \__enumext_stop_item_vii:
5172   {
5173       \__enumext_endminipage:
5174     \__enumext_stop_list_tag:n {enumext*}
5175     \hbox_set_end:
```

Here we will reduce the *warnings* a bit by setting the value of \hbadness to 10000, print \item and *"item content"* from the *horizontal box*,.

```
    \int_set:Nn \hbadness { 10000 }
    \box_use_drop:N \l__enumext_item_text_vii_box
```

Finally apply the *vertical space* between rows set by itemsep key passed to \parsep using \par\noindent and *horizontal space* between columns set by columns-sep key using \skip_horizontal:N.

```
    \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
      }
      {
        \skip_horizontal:N \l__enumext_columns_sep_vii_dim
      }
  }
```

(*End of definition for* __enumext_start_item_vii:w *and* __enumext_stop_item_vii:.)

__enumext_remove_extra_parsep_vii:

Remove the extra *vertical space* equal to \parsep=\itemsep when the total number of \item is divisible by the number of \item in the last row of the environment. Here the use of \unskip or \removelastskip fails and does not obtain the expected result, using \vspace is the option and in this case, we can use a simplified version since we are always in ⟨*vertical mode*⟩.

```
\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 }
      {
        \para_end:
        \skip_vertical:n { -\l__enumext_itemsep_vii_skip }
        \skip_vertical:N \c_zero_skip
        \int_gzero:N \g__enumext_item_count_all_vii_int
      }
  }
```

(*End of definition for* __enumext_remove_extra_parsep_vii:.)

As we don't want our check to be executed check-ans by levels but on the complete list, we will take it out of the enumext* environment using the *"hook"* function __enumext_after_env:nn.

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

13.48 The environment keyans*

keyans*

The implementation of keyans* environment is the similar as that used by the enumext* environment except for the __enumext_check_starred_cmd:n function added in the second part.

```
\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:
      }
    \setcounter { enumXviii } { \int_eval:n { \int_use:c { l__enumext_start_viii_int } - 1 } }
    \IfDocumentMetadataT { \tag_suspend:n {keyans*} }
    \__enumext_starred_columns_set_viii:
    \item[] \scan_stop:
    \cs_set_eq:NN \__enumext_stop_item_tmp_viii: \__enumext_first_item_tmp_viii:
    \cs_set_eq:NN \item \__enumext_start_item_tmp_viii:
    \ignorespaces
```

```
  }
  {
    \IfDocumentMetadataT { \tag_struct_end:n {tag=text-unit} }
    \__enumext_stop_item_tmp_viii:
    \__enumext_remove_extra_parsep_viii:
    \__enumext_check_starred_cmd:n { item }
    \__enumext_after_list_viii:
  }
```

(*End of definition for* keyans*. This function is documented on page 16.*)

__enumext_safe_exec_viii:

The __enumext_safe_exec_viii: function will first check if the save-ans key is active and only when this is true the environment will be available, it will increment the value of \l__enumext_keyans_level_h_int and return an error message when we are nesting the environment, then it will call the __enumext_keyans_name_and_start: function in charge of saving the name of the environment and the line it is running on, then it will check if we are trying to nest keyans* in enumext* returning an error and we will set \l__enumext_starred_bool to true, finally we will check if we are within the appropriate level within the enumext environment.

```
\cs_new_protected:Nn \__enumext_safe_exec_viii:
  {
    \bool_if:NF \l__enumext_store_active_bool
      {
        \msg_error:nnnn { enumext } { wrong-place }{ keyans* }{ save-ans }
      }
    \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_name_and_start:
    \bool_if:NT \l__enumext_starred_bool
      {
        \msg_error:nnn { enumext } { nested-horizontal } { enumext* }
      }
    \bool_set_true:N \l__enumext_starred_bool
    % 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_viii: 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:.)

13.48.1 The command \item in keyans*

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

__enumext_first_item_tmp_viii:

The __enumext_first_item_tmp_viii: function will remove horizontal space equal to \labelwidth plus \labelsep to the left of the *"first"* \item in the environment at the point of execution of this function, where it is equal to the __enumext_stop_item_tmp_viii: function inside the environment body definition.

```
\cs_new_protected_nopar:Nn \__enumext_first_item_tmp_viii:
  {
    \skip_horizontal:n
      {
        -\l__enumext_labelwidth_viii_dim - \l__enumext_labelsep_viii_dim
      }
    \ignorespaces
  }
```

(*End of definition for* __enumext_first_item_tmp_viii:.)

__enumext_start_item_tmp_viii:
__enumext_item_peek_args_viii:
__enumext_joined_item_viii:w
__enumext_standar_item_viii:w

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:
  }
```

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) }
  }
```

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 }
  }
```

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
    \bool_set_false:N \l__enumext_item_wrap_key_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 ] \ignorespaces
      }
  }
```

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

__enumext_starred_item_viii:w
__enumext_starred_item_viii_aux_i:w
__enumext_starred_item_viii_aux_ii:w
__enumext_keyans_starred_item_star:

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_item_wrap_key_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_viii_tl
          {
            \tl_put_right:NV \l__enumext_store_current_label_tl \l__enumext_store_keyans_item_opt
            \tl_put_right:Nn \l__enumext_store_current_label_tl { #1 }
          }
        \tl_set:Nn \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 ] \ignorespaces
  }
```

The function __enumext_keyans_starred_item_star: 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, mark-sep and save-ref are implemented.

```
\cs_new_protected:Nn \__enumext_keyans_starred_item_star:
  {
    \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:Nn \l__enumext_store_current_label_tl { \item }
```

```
    \__enumext_keyans_addto_seq_link:
    \int_gincr:N \g__enumext_check_starred_cmd_int
    \dim_compare:nNnT { \l__enumext_mark_sym_sep_viii_dim } = { \c_zero_dim }
      {
        \dim_set:Nn \l__enumext_mark_sym_sep_viii_dim { \l__enumext_labelsep_viii_dim }
      }
    \bool_if:NT \l__enumext_show_answer_bool
      {
        \tl_set_eq:NN \l__enumext_mark_answer_sym_tl \l__enumext_mark_answer_sym_viii_tl
        \str_set_eq:NN \l__enumext_mark_position_str \l__enumext_mark_position_viii_str
        \__enumext_print_keyans_box:NN
          \l__enumext_labelwidth_viii_dim \l__enumext_mark_sym_sep_viii_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:
          }
        \str_set_eq:NN \l__enumext_mark_position_str \l__enumext_mark_position_viii_str
        \__enumext_print_keyans_box:NN
          \l__enumext_labelwidth_viii_dim \l__enumext_mark_sym_sep_viii_dim
      }
  }
```

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

`\__enumext_keyans_wraper_label_viii:n`
`\__enumext_fake_make_label_viii:n`

The implementation at this is very similar to that of the `enumext*` environment.

```
\cs_new_protected:Npn \__enumext_keyans_wraper_label_viii:n #1
  {
    \bool_lazy_all:nT
      {
        { \bool_if_p:N \l__enumext_wrap_label_viii_bool         }
        { \bool_if_p:N \l__enumext_show_answer_bool             }
        { \bool_if_p:N \l__enumext_item_wrap_key_bool           }
        { \cs_if_exist_p:N \__enumext_keyans_wrapper_item_viii:n }
      }
      {
        \cs_set_eq:NN
          \__enumext_wrapper_label_viii:n \__enumext_keyans_wrapper_item_viii:n
      }
    \bool_if:NTF \l__enumext_wrap_label_viii_bool
      {
        \__enumext_wrapper_label_viii:n {#1}
      }
      { #1 }
  }
\cs_new_protected_nopar:Npn \__enumext_fake_make_label_viii:n #1
  {
    \legacy_if:nT { @noitemarg }
      {
        \legacy_if_set_false:n { @noitemarg }
        \legacy_if:nT { @nmbrlist }
          {
            \refstepcounter{enumXviii}
          }
      }
    \bool_if:NT \l__enumext_item_starred_viii_bool
      {
        \__enumext_keyans_starred_item_star:
      }
    \makebox[ \l__enumext_labelwidth_viii_dim ][ \l__enumext_align_label_viii_str ]
      {
        \tl_use:N \l__enumext_label_font_style_viii_tl
```

```
5420          \__enumext_keyans_wraper_label_viii:n {#1}
5421        }
5422      \skip_horizontal:N \l__enumext_labelsep_viii_dim \ignorespaces
5423    }
```

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

13.48.2 Real definition of `\item` in keyans*

`\__enumext_start_item_viii:w` `\__enumext_stop_item_viii:`

The implementation at this is very similar to that of the enumext* environment.

```
5424 \cs_new_protected_nopar:Npn \__enumext_start_item_viii:w [#1]
5425   {
5426     \cs_set_eq:NN \__enumext_stop_item_tmp_viii: \__enumext_stop_item_viii:
5427     \hbox_set_to_wd:Nnw \l__enumext_item_text_viii_box
5428       {
5429         \l__enumext_joined_width_viii_dim
5430         + \l__enumext_labelwidth_viii_dim
5431         + \l__enumext_labelsep_viii_dim
5432       }
5433       \__enumext_renew_footnote_starred:
5434       \__enumext_start_list_tag:n {keyans*}
5435       \__enumext_fake_make_label_viii:n {#1}
5436       \__enumext_stop_start_list_tag:
5437       \__enumext_minipage:w [ t ]{ \l__enumext_joined_width_viii_dim  }
5438         \dim_set_eq:NN \parindent \l__enumext_listparindent_viii_dim
5439         \skip_set_eq:NN \parskip \l__enumext_parsep_viii_skip
5440         \__enumext_unskip_unkern:
5441         \__enumext_unskip_unkern:
5442         \skip_horizontal:n { -\l__enumext_listparindent_viii_dim } \ignorespaces
5443         \tl_use:N \l__enumext_fake_item_indent_viii_tl
5444         \bool_if:NT \l__enumext_item_starred_viii_bool
5445           {
5446             \__enumext_keyans_show_item_opt_viii:
5447           }
5448         \tl_use:N \l__enumext_after_list_args_viii_tl
5449    }
5450 \cs_new_protected_nopar:Nn \__enumext_stop_item_viii:
5451   {
5452       \__enumext_endminipage:
5453     \__enumext_stop_list_tag:n {keyans*}
5454     \hbox_set_end:
5455     \int_set:Nn \hbadness { 10000 }
5456     \box_use_drop:N \l__enumext_item_text_viii_box
5457     \int_compare:nNnTF
5458       { \l__enumext_item_column_pos_viii_int } = { \l__enumext_columns_viii_int }
5459       {
5460         \par\noindent
5461         \int_zero:N \l__enumext_item_column_pos_viii_int
5462       }
5463       {
5464         \skip_horizontal:N \l__enumext_columns_sep_viii_dim
5465       }
5466   }
```

(*End of definition for* `\__enumext_start_item_viii:w` *and* `\__enumext_stop_item_viii:`.)

`\__enumext_remove_extra_parsep_viii:`

The implementation at this is very similar to that of the enumext* environment.

```
5467 \cs_new_protected:Nn \__enumext_remove_extra_parsep_viii:
5468   {
5469     \int_compare:nNnT
5470       {
5471         \int_mod:nn
5472           { \g__enumext_item_count_all_viii_int }
5473           { \l__enumext_columns_viii_int }
5474       }
5475       =
5476       { 0 }
5477       {
5478         \para_end:
5479         \skip_vertical:n { -\l__enumext_itemsep_viii_skip }
5480         \skip_vertical:N \c_zero_skip
5481         \int_gzero:N \g__enumext_item_count_all_viii_int
```

```
5482        }
5483    }
```

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

13.49 The command `\getkeyans`

`\getkeyans`
`\__enumext_getkeyans_aux:n`
`\__enumext_getkeyans:nn`

The `\getkeyans` command takes a *mandatory argument* of the form {⟨*store name* : *position*⟩}. Retrieve a *"single content"* stored by `\anskey`, `\anspic*` and `\item*` and `anskey*` 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} } }
  }
```

The internal function `\__enumext_getkeyans_aux:n` is in charge of *splitting* the *mandatory 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} }
  }
```

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:cTF { g__enumext_#1_prop }
      {
        \prop_item:cn { g__enumext_#1_prop }{#2}
      }
      {
        \msg_error:nnn { enumext } { undefined-storage-anskey } {#1}
      }
  }
```

(*End of definition for* `\getkeyans`, `\__enumext_getkeyans_aux:n`, *and* `\__enumext_getkeyans:nn`. *This function is documented on page 19.*)

13.50 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*⟩`]`.

```
\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  = { labelwidth=0pt, labelsep=0.3333em, itemindent=0pt, list-offset=0pt,
                            rightmargin=0pt, listparindent=0pt, 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  = { labelwidth=0pt, labelsep=0.3333em, itemindent=0pt, list-offset=0pt,
```

```
                            rightmargin=0pt, listparindent=0pt, 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  = { labelwidth=0pt, labelsep=0.3333em, itemindent=0pt, list-offset=0pt,
                            rightmargin=0pt, listparindent=0pt,  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  = { labelwidth=0pt, labelsep=0.3333em, itemindent=0pt, list-offset=0pt,
                            rightmargin=0pt, listparindent=0pt, 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  = { labelwidth=0pt, labelsep=0.3333em, itemindent=0pt, list-offset=0pt,
                            rightmargin=0pt, listparindent=0pt, 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  = { labelwidth=0pt, labelsep=0.3333em, itemindent=0pt, list-offset=0pt,
                            rightmargin=0pt, listparindent=0pt, 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`
`\__enumext_printkeyans:nnn`

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:
      \bool_set_true:N \l__enumext_print_keyans_cmd_bool
      \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 }
      \bool_set_false:N \l__enumext_print_keyans_cmd_bool
    \group_end:
  }
```

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 }
          {
```

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*`, we set `\l__enumext_base_line_fix_bool` and `\l__enumext_print_keyans_star_bool` to true for *baseline correction*, open the `enumext*` environment passing the *optional argument* and map the *sequence*, then set `\l__enumext_base_line_fix_bool` and `\l__enumext_print_keyans_star_bool` to false.

```
            \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
                    \bool_set_true:N \l__enumext_base_line_fix_bool
                    \bool_set_true:N \l__enumext_print_keyans_star_bool
                    \begin{enumext*}[#2]
                      \seq_map_inline:cn { g__enumext_#3_seq } { ##1 }
                    \end{enumext*}
                    \bool_set_false:N \l__enumext_base_line_fix_bool
                    \bool_set_false:N \l__enumext_print_keyans_star_bool
                  }
              }
```

Otherwise it will open the environment `enumext` passing the *optional argument* to the *"first level"* then map the *sequence*.

```
              {
                \begin{enumext}[#2]
                  \seq_map_inline:cn { g__enumext_#3_seq } { ##1 }
                \end{enumext}
              }
          }
      }
      {
        \msg_error:nnn { enumext } { undefined-storage-anskey } {#3}
      }
  }
```

(*End of definition for* `\printkeyans` *and* `\__enumext_printkeyans:nnn`*. This function is documented on page* 20.)

13.51 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*` and `enumext` environments so as not to capture ⟨*keys*⟩ that complicate us.

`\__enumext_filter_level:n`
`\__enumext_filter_level_key:n`
`\__enumext_filter_level_pair:nn`

The function `\__enumext_filter_level:n` will be in charge of filtering the ⟨*keys*⟩ passed to the `enumext` and `enumext*` environments.

```
\cs_new:Npn \__enumext_filter_level:n #1
  {
    \use:e
      {
        \keyval_parse:NNn
          \__enumext_filter_level_key:n
          \__enumext_filter_level_pair:nn {#1}
      }
  }
```

The function `\__enumext_filter_level_key:n` will be responsible for filtering the ⟨*keys*⟩ that are passed *"without value"* by excluding the keys `resume*`, `reset` and `reset*` passed to the `enumext` and `enumext*` environments.

```
\cs_new:Npn \__enumext_filter_level_key:n #1
  {
    \str_case:nnF {#1}
      {
        { resume* } {} { reset } {} { reset* } {}
      }
      { , { \exp_not:n {#1} } }
  }
```

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

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

(*End of definition for* `\__enumext_filter_level:n`*,* `\__enumext_filter_level_key:n`*, and* `\__enumext_filter_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_level:n {#1}
                            }
                        },
    enumext-2 .code:n = {
                           \keys_set:ne { enumext / level-2 }
                            {
                              \__enumext_filter_level:n {#1}
                            }
                        },
    enumext-3 .code:n = {
                          \keys_set:ne { enumext / level-3 }
                            {
                              \__enumext_filter_level:n {#1}
                            }
                        },
    enumext-4 .code:n = {
                          \keys_set:ne { enumext / level-4 }
                            {
                              \__enumext_filter_level:n {#1}
                            }
                        },
    enumext*  .code:n = {
                          \keys_set:ne { enumext / enumext* }
                           {
                             \__enumext_filter_level:n {#1}
                           }
                        },
    keyans    .code:n = { \keys_set:nn { enumext / keyans  } {#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`.

`\__enumext_set_parse:n`
`\__enumext_set_error:nn`

```
\NewDocumentCommand \setenumext { O{enumext,1} +m }
  {
    \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} }
      }
  }
```

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* \setenumext, __enumext_set_parse:n, *and* __enumext_set_error:nn. *This function is documented on page 6.*)

13.52 The command \setenumextmeta

The command \setenumextmeta will be responsible for adding new *"meta-keys"* for the enumext and enumext* environments. The implementation code was given by Jonathan P. Spratte (@Skillmon) answer in Simplify syntax for command that adds .meta key to existing keys (l3keys).

\setenumextmeta
__enumext_key_set_meta:nnn
__enumext_key_def_meta:nnn
__enumext_key_def_meta:Vnn

First we will create a ⟨*keys*⟩ of type .code:n for *"all levels"* of the enumext environment.

```
\int_step_inline:nn { 4 }
  {
    \keys_define:nn { enumext }
      {
         #1 .code:n = \str_set:Nn \l__enumext_meta_path_str { level-#1 }
        ,#1 .value_forbidden:n = true
      }
  }
```

And now we define the ⟨*keys*⟩ for the environments using .code:n for the enumext environment and .meta:n for the enumext* environment.

```
\clist_map_inline:nn { enumext }
  {
    \keys_define:nn { enumext }
      {
         #1  .code:n = % ignored for now, might do something useful in the future
        ,#1  .value_forbidden:n = true
        ,#1* .code:n = \str_set:Nn \l__enumext_meta_path_str { #1* }
        ,#1* .value_forbidden:n = true
      }
  }
\keys_define:nn { enumext }
  {
     * .meta:n = enumext*
    ,* .value_forbidden:n = true
  }
```

Now we create the user command taking care that unknown cannot be passed as an argument.

```
\NewDocumentCommand \setenumextmeta { s O{enumext,1} m +m }
  {
    \str_if_eq:eeTF { \tl_trim_spaces:n {#3} } { unknown }
      { \msg_error:nn { enumext } { prohibited-unknown } }
      {
        \bool_if:nTF {#1}
          {
            \int_step_inline:nn { 4 }
              { \__enumext_key_set_meta:nnn { enumext, ##1 } {#3} {#4} }
            \__enumext_key_set_meta:nnn { enumext* } {#3} {#4}
          }
```

```
5745            { \__enumext_key_set_meta:nnn {#2} {#3} {#4} }
5746        }
5747    }
```

The internal functions `\__enumext_key_set_meta:nnn` and `\__enumext_key_def_meta:nnn` will check the *optional argument* and create the *"meta-key"*.

```
5748 \cs_new_protected:Npn \__enumext_key_set_meta:nnn #1
5749   {
5750     \keys_set:nn { enumext } {#1}
5751     \__enumext_key_def_meta:Vnn \l__enumext_meta_path_str
5752   }
5753 \cs_new_protected:Npn \__enumext_key_def_meta:nnn #1#2#3
5754   {
5755     \bool_lazy_or:nnTF
5756       { \keys_if_exist_p:nn { enumext / #1} {#2} }
5757       { \keys_if_exist_p:nn { enumext / enumext* } {#2} }
5758       { \msg_error:nnn { enumext } { already-defined } {#2} }
5759       {
5760         \keys_define:nn { enumext / #1 }
5761           {
5762             #2 .meta:n = {#3},
5763             #2 .value_forbidden:n = true
5764           }
5765       }
5766   }
5767 \cs_generate_variant:Nn \__enumext_key_def_meta:nnn { V }
```

(*End of definition for* `\setenumextmeta`, `\__enumext_key_set_meta:nnn`, *and* `\__enumext_key_def_meta:nnn`. *This function is documented on page* 6.)

13.53 The command `\foreachkeyans`

The command `\foreachkeyans` will execute a *loop* over the *prop list* and return its contents. The implementation code is adapted from the answer provided by Enrico Gregorio (`@egreg`) in Expand a `.cs` defined by key inside the function.

`\foreachkeyans`
`\__enumext_parse_foreach_keys:nn`
`\__enumext_parse_foreach_keys:n`
`\__enumext_foreach_keyans:nn`
`\__enumext_foreach_add_body:n`

We define a set of ⟨*keys*⟩ for command and we will save the default values of these in `\g__enumext_foreach_default_keys_tl` to avoid the use of group.

```
5768 \keys_define:nn { enumext / foreach }
5769   {
5770     before  .tl_set:N  = \l__enumext_foreach_before_tl,
5771     before  .value_required:n = true,
5772     after   .tl_set:N  = \l__enumext_foreach_after_tl,
5773     after   .value_required:n = true,
5774     start   .int_set:N = \l__enumext_foreach_start_int,
5775     start   .value_required:n = true,
5776     stop    .int_set:N = \l__enumext_foreach_stop_int,
5777     stop    .value_required:n = true,
5778     step    .int_set:N = \l__enumext_foreach_step_int,
5779     step    .value_required:n = true,
5780     wrapper .cs_set_protected:Np = \__enumext_foreach_wrapper:n #1,
5781     wrapper .value_required:n = true,
5782     sep     .tl_set:N  = \l__enumext_foreach_sep_tl,
5783     sep     .value_required:n = true,
5784     unknown .code:n    = { \__enumext_parse_foreach_keys:n {#1} }
5785   }
5786 \keys_precompile:nnN { enumext / foreach }
5787   {
5788     before={},after={},start=1,step=1,stop=0,wrapper=#1,sep={; }
5789   }
5790   \l__enumext_foreach_default_keys_tl
```

Functions for handling unknown ⟨*keys*⟩.

```
5791 \cs_new_protected:Npn \__enumext_parse_foreach_keys:nn #1#2
5792   {
5793     \tl_if_blank:nTF {#2}
5794       {
5795         \msg_error:nnn { enumext } { for-key-unknown } {#1}
5796       }
5797       {
5798         \msg_error:nnnn { enumext } { for-key-value-unknown } {#1} {#2}
5799       }
5800   }
```

```
\cs_new_protected:Npn \__enumext_parse_foreach_keys:n #1
  {
    \exp_args:NV \__enumext_parse_foreach_keys:nn \l_keys_key_str {#1}
  }
```

We create the command.

```
\NewDocumentCommand \foreachkeyans { +O{} m }
  {
    \__enumext_foreach_keyans:nn {#1} {#2}
  }
```

Finally the internal functions `\__enumext_foreach_keyans:nn` and `\__enumext_foreach_add_body:n` will loop through the prop list and print the contents.

```
\cs_new_protected:Npn \__enumext_foreach_keyans:nn #1 #2
  {
    \tl_use:N \l__enumext_foreach_default_keys_tl
    \keys_set:nn { enumext / foreach } {#1}
    \tl_set:Nn \l__enumext_foreach_name_prop_tl {#2}
    \prop_if_exist:cF { g__enumext_#2_prop }
      {
        \msg_error:nnn { enumext } { undefined-storage-anskey } {#2}
      }
    \int_compare:nNnT { \l__enumext_foreach_stop_int } = { 0 }
      {
        \int_set:Nn \l__enumext_foreach_stop_int
          { \prop_count:c { g__enumext_#2_prop } }
      }
    \seq_clear:N \l__enumext_foreach_print_seq
    \int_step_function:nnnN
      { \l__enumext_foreach_start_int }
      { \l__enumext_foreach_step_int }
      { \l__enumext_foreach_stop_int }
      \__enumext_foreach_add_body:n
      \seq_use:NV \l__enumext_foreach_print_seq \l__enumext_foreach_sep_tl
  }
\cs_new_protected:Npn \__enumext_foreach_add_body:n #1
  {
    \seq_put_right:Ne \l__enumext_foreach_print_seq
      {
        \exp_not:V \l__enumext_foreach_before_tl
        \__enumext_foreach_wrapper:n
          {
            \prop_item:cn { g__enumext_ \l__enumext_foreach_name_prop_tl _prop }{#1}
          }
        \exp_not:V \l__enumext_foreach_after_tl
      }
  }
```

(*End of definition for* `\foreachkeyans` *and others. This function is documented on page 19.*)

13.54 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~environment~'#1'~is~reserved~by ~\\
    'enumext'~package,~It~is~already~defined.
  }
  {
    The~environment~'#1'~is~defined~internally ~
    for~the~'save-ans'~key~with~save-ans~key~active.~See~documentation.\\
  }
\msg_new:nnn { enumext } { anskey-env-nested }
  {
    The~#1~'#2'~can't~be~nested~\msg_line_context:.
  }
```

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~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:nnn { enumext } { overwrite-file }
  {
    Overwriting~file~'#1'.
  }
\msg_new:nnn { enumext } { writing-file }
  {
    Writing~file~'#1'.
  }
\msg_new:nnn { enumext } { not-writing }
  {
    File~`#1'~already~exists.~Not~writing.
  }
```

Messages used by keyans, 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.
  }
\msg_new:nnn { enumext } { keyanspic-item-cmd }
  {
    Can't~use ~\c_backslash_str item~in~keyanspic~\msg_line_context:.
  }
\msg_new:nnnn { enumext } { keyans-unknown-key }
  {
    The~key~'#1'~is~unknown~by~environment~
    '\l__enumext_envir_name_tl'~and~is~being~ignored.
  }
  {
    The~environment~'\l__enumext_envir_name_tl'~does~not
   ~have~a~key~called ~'#1'.\\
    Check~that~you~have~spelled~the~key~name~correctly.
  }
\msg_new:nnnn { enumext } { keyans-unknown-key-value }
  {
    The~key~'#1=#2'~is~unknown~by~environment ~
    '\l__enumext_envir_name_tl'~and~is~being~ignored.
  }
  {
    The~environment~'\l__enumext_envir_name_tl'~does~not
   ~have~a~key~called ~'#1'.\\
    Check~that~you~have~spelled~the~key~name~correctly.
  }
```

Message used by unknown ⟨*keys*⟩ in `enumext*`. environment.

```
\msg_new:nnnn { enumext } { starred-unknown-key }
  {
    The~key~'#1'~is~unknown~by~environment~
    '\l__enumext_envir_name_tl'~and~is~being~ignored.
  }
  {
    The~environment~'\l__enumext_envir_name_tl'~does~not
   ~have~a~key~called ~'#1'.\\
    Check~that~you~have~spelled~the~key~name~correctly.
  }
\msg_new:nnnn { enumext } { starred-unknown-key-value }
  {
    The~key~'#1=#2'~is~unknown~by~environment ~
    '\l__enumext_envir_name_tl'~and~is~being~ignored.
  }
  {
    The~environment~'\l__enumext_envir_name_tl'~does~not
   ~have~a~key~called ~'#1'.\\
    Check~that~you~have~spelled~the~key~name~correctly.
  }
```

Message used by unknown ⟨*keys*⟩ in `enumext` environment.

```
\msg_new:nnnn { enumext } { standar-unknown-key }
  {
    The~key~'#1'~is~unknown~by~environment~'\l__enumext_envir_name_tl' \c_space_tl
   ~on~level~\int_use:N \l__enumext_level_int \c_space_tl and~is~being~ignored.
  }
  {
    The~environment~'\l__enumext_envir_name_tl'~does~not
   ~have~a~key~called ~'#1'~on~level~\int_use:N \l__enumext_level_int.\\
    Check~that~you~have~spelled~the~key~name~correctly.
  }
\msg_new:nnnn { enumext } { standar-unknown-key-value }
  {
    The~key~'#1=#2'~is~unknown~by~environment~'\l__enumext_envir_name_tl' \c_space_tl
   ~on~level~\int_use:N \l__enumext_level_int \c_space_tl and~is~being~ignored.
  }
  {
    The~environment~'\l__enumext_envir_name_tl'~does~not
   ~have~a~key~called ~'#1'~on~level~\int_use:N \l__enumext_level_int.\\
    Check~that~you~have~spelled~the~key~name~correctly.
  }
```

Message used by unknown ⟨*keys*⟩ in `\foreachkeyans`.

```
\msg_new:nnnn { enumext } { for-key-unknown }
  { The~key~'#1'~is~unknown~by~'\c_backslash_str foreachkeyans'~and~is~being~ignored.}
  {
    The~command~'\c_backslash_str foreachkeyans'~does~not~have~a~key~called~'#1'.\\
    Check~that~you~have~spelled~the~key~name~correctly.
  }
\msg_new:nnnn { enumext } { for-key-value-unknown }
  { The~key~'#1=#2'~is~unknown~by~'\c_backslash_str foreachkeyans'~and~is~being~ignored. }
  {
    The~command~'\c_backslash_str foreachkeyans'~does~not~have~a~key~called~'#1'.\\
    Check~that~you~have~spelled~the~key~name~correctly.
  }
```

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'.
  }
\msg_new:nnn { enumext } { wrong-miniright-starred }
  {
    Can't~use ~\c_backslash_str miniright~in~starred~environments~\msg_line_context:.
  }
\msg_new:nnn { enumext } { many-miniright-used }
  {
    Can't~use ~\c_backslash_str miniright~more~than~once~ \msg_line_context:.
  }
```

Messages used by `\setenumextmeta` command.

```
\msg_new:nnn { enumext } { already-defined }
  {
    The~key~'#1'~is~already~defined~\msg_line_context:.
  }
\msg_new:nnn { enumext } { prohibited-unknown }
  {
    The~name~'unknown'~can't~be~chosen~ for~a~meta~key~\msg_line_context:.
  }
```

Messages used by `enumext*` and `keyans*` environments.

```
\msg_new:nnn { enumext } { nested }
  {
    The~environment~\l__enumext_envir_name_tl \c_space_tl can't~be~nested~\msg_line_context:.
  }
\msg_new:nnn { enumext } { nested-horizontal }
  {
    The~environment~\l__enumext_envir_name_tl \c_space_tl can't~be~nested~in~'#1'~ \msg_line_conte
  }
\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:.
  }
```

Messages used by `resume` key.

```
\msg_new:nnn { enumext } { unknown-series-starred }
  {
    The~series~'#1'~for~the~resume~key~does~not~exist~in~the~
    ~enumext*~environment~ \msg_line_context:.
  }
\msg_new:nnn { enumext } { unknown-series-standar }
  {
    The~series~'#1'~for~the~resume~key~does~not~exist~at~level~\int_use:N \l__enumext_level_int
    \c_space_tl of~enumext~environment~ \msg_line_context:.
  }
\msg_new:nnnn { enumext } { out-of-range }
  { The~number~must~be~exactly~1,~2,~3~or~4. }
  { Received:~'#1'. }
```

13.55 Finish package

Finish package implementation.

```
\file_input_stop:
⟨/package⟩
```

14 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.

D

E

Environments provide by enumext:

K

L

M

N

O

P

R

S

T

U

V

W