

enumext

ENUMERATE EXERCISE SHEETS

V1.0 2024-04-22^{*}

©2024 by Pablo González[†]

CTAN: <https://www.ctan.org/pkg/enumext>

 <https://github.com/pablgonz/enumext>

Abstract

This package provides “*enumerated list*” environments for creating “*simple exercise sheets*” along with “*multiple choice questions*”, storing the *(answers)* to these in memory using the **multicol** package and the **l3seq** and **l3prop** modules.

Contents

1 Introduction	2	4 The storage system	9
1.1 Description and usage	3	4.1 Keys for storage	10
1.2 The concept of left margin	3	4.2 Keys for internal label and ref	10
1.3 User interface	3	4.3 Keys for check answers	10
1.3.1 Internal counters	3	4.4 The command <code>\anskey</code>	10
1.3.2 Support for multicol	4	4.5 The environment keyans	11
1.3.3 Support for minipage	4	4.5.1 The <code>\item*</code> in keyans	11
1.3.4 The <code>\label</code> and <code>\ref</code> system	4	4.6 The environment keyanspic	12
1.3.5 Support for <code>\footnote</code>	4	4.6.1 The command <code>\anspic</code>	12
2 The environment <code>enumext</code>	4	4.7 Printing stored content	13
2.1 The <code>\item*</code> in <code>enumext</code>	5	4.7.1 The command <code>\getkeyans</code>	13
2.1.1 Keys for <code>\item*</code> in <code>enumext</code>	5	4.7.2 The command <code>\printkeyans</code>	13
3 The command <code>\setenumext</code>	5	5 Full examples	14
3.1 Keys for label and ref	6	6 The way of non-enumerated lists	16
3.2 Keys for spaces	6	7 References	18
3.2.1 Vertical spaces	7	8 Change history	18
3.2.2 Horizontal spaces	8	9 Index of Documentation	19
3.3 Keys for add code	8	10 Implementation	21
3.4 Keys for start and resume	9	11 Index of Implementation	94
3.5 Keys for multicol	9		
3.6 Keys for minipage	9		
3.6.1 The command <code>\miniright</code>	9		

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 \TeX 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 multicol, vertical alignment

License and Requirements

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

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

^{*}This file describes a documentation for v1.0, last revised 2024-04-22.

[†]E-mail: pablgonz@educarchile.cl.

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) \LaTeX 2e 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) \LaTeX 2e 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

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

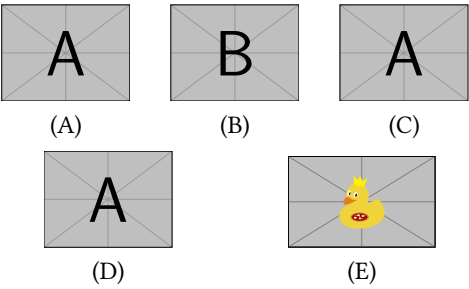
(B) value

(C) value

(D) value

(E) value

4. Question with image and label below:



5. Question with image on left side:

- (A) value

(B) value

(C) value

(D) correct

(E) value
-

Where what we are interested in the `<label>` and a “short note” that we leave as an explanation, and then print them:

1. (B) $x = 5$

2. (D)

3. (C) some note
4. (B)

5. (D) “other note”

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

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

1.1 Description and usage

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

- This package is NOT intend to replace the `enumerate` environment nor replace the powerful `enumitem`[5], the approach is intended to work without hindering either of them. This package can be used with `xelatex`, `lualatex`, `pdflatex` and the classical `latex>dvips>ps2pdf` and is present in \TeX Live and \MiKTeX , use the package manager to install. For manual installation, download `enumext.zip` and unzip it, run `lualatex enumext.dtx` and move all files to appropriate locations, then run `mktxlsr`. To produce the documentation run `lualatex enumext.dtx` two times.

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

The package is loaded in the usual way:

```
\usepackage{enumext}
```

1.2 The concept of left margin

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

Usually we don’t want the `list` to go beyond the left margin of the page, but since these four values are related, that causes a problem. The `enumitem`[5] package adds the `\labelindent` parameter to solve some of these problems. A simplified representation of this in the figure 1.

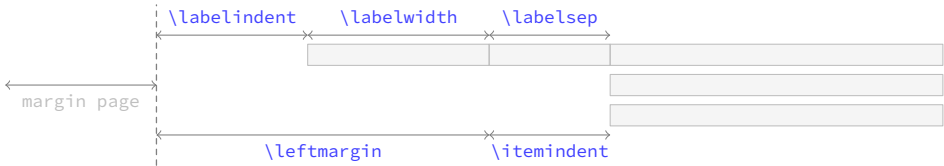


Figure 1: Representation of horizontal lengths in `enumitem`.

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

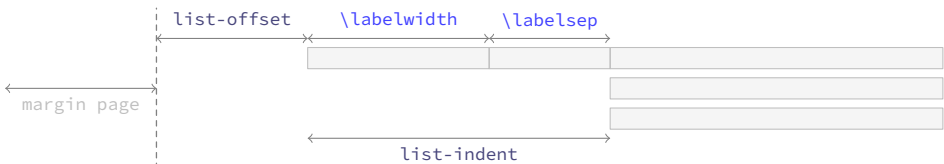


Figure 2: Representation of horizontal lengths concept in `enumext`.

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



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

1.3 User interface

The user interface consists in `enumext`, `enumext*`, `keyans`, `keyans*` and `keyanspic` environments, `\anskey`, `\item*` and `\anspic*` commands to \langle stored content \rangle , `\getkeyans` command to get the individual \langle stored content \rangle , `\printkeyans` to print all \langle stored content \rangle , `\miniright` for `minipage` and `\setenumext` to config all [`\key = val`] options.

1.3.1 Internal counters

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

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

1.3.2 Support for multicols

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

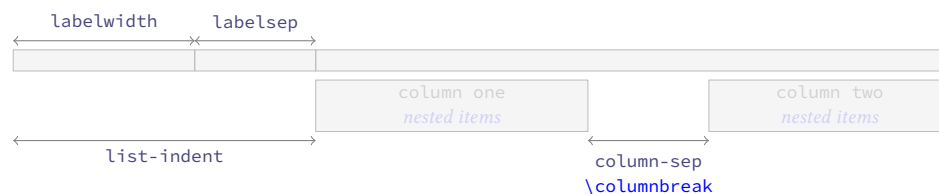


Figure 4: Representation of the two column output for a nested level in `enumext` environment.

The “non starred” version of the `multicols` environment is always used together with the `\raggedcolumns` command and is controlled by `columns` and `columns-sep` keys. The environment is available for all nesting levels, and can can together with the `mini-env` key. If you need to force a start a new column `\columnbreak` must be used (see §3.5).

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

1.3.3 Support for minipage

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



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

The `minipage` environments (left and right) is always used with “aligned on top” [`t`], the `minipage` environment on the “right side” always starts with `\centering`. It can be used at all nesting levels and is controlled by `mini-env` and `mini-sep` keys. In order to switch from the “left” side `minipage` environment to the “right” side one must use the command `\miniright` (see §3.6).

1.3.4 The \label and \ref system

This package provides a user interface like the `enumitem`[5] package to customize the references which is activated by the `ref` key (§3.1), the standard `\label` and `\ref` commands work as usual. It also provides an “internal reference” system for the “stored content” by means of the key `store-ref` (§4.2) when the key `save-ans`(§4.1) is active.

- The implementation of `\label` and `\ref` together with the `store-ref` key are compatible with the `hyperref`[7] package.

1.3.5 Support for \footnote

This package provides an internal implementation for the `\footnote` command which is compatible with the `hyperref` package, but, it will not produce the expected links, and when using the `mini-env` key or the starred environments `enumext*` and `keyans*` the output will look like the classic way they are displayed in the `minipage` environment.

The best way to solve this is to use Jean-François Burnol `footnotehyper`[8] package, it will support keeping the links if `hyperref` is loaded with the `hyperfootnotes=true` option (default) and will show the output numbered at the bottom of the page (as opposed to how it is displayed in the `minipage` environment). The way to load it is as follows:

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

2 The environment enumext

```
enumext \begin{enumext} [⟨keyval list⟩]
enumext* \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*}
```

The `enumext` is an “*enumerated list*” environment that works in the same way as the standard `enumerate` environment provided by L^AT_EX, `\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

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.

```
\begin{enumext}
  \item This text is in the first level.
  \begin{enumext}
    \item This text is in the second level.
    \begin{enumext}
      \item This text is in the third level.
      \begin{enumext}
        \item This text is in the fourth level.
      \end{enumext}
    \end{enumext}
  \end{enumext}
  \item[X] This text is in the first level.
  \item* This text is in the first level.
\end{enumext}
```

2.1 The \item* in enumext

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

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

The *starred version* ‘★’ 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` environment is NOT the same as in the `keyans` environment.

2.1.1 Keys for \item* in enumext

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

`item-pos*` = {`⟨rigid length | dim expression⟩`} 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*.

3 The command \setenumext

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

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

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

- It should be kept in mind that using any *key* that sets a *rubber or rigid lengths* for vertical or horizontal space on a level will influence the vertical and horizontal space for *inners levels* and *keyans* and *keyanspic* 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 `\dimexpr` or `\dimeval` to perform calculations.

3.1 Keys for label and ref

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

Sets the *label* that will be printed at the *current level*. The default value for first level are `\arabic*`, for second level are `(\alph*)`, for third level are `\roman*`, and for fourth level are `\Alph*`.

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

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

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

- Internally, it renews the command associated with each counter when it is executed, i.e., `\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{enumxi}` to indicate the count of the first level instead of using `\theenumxi`.

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

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

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

Sets the *width* of the box containing the current *label* set by `label` key. Internally sets the value of `\labelwidth` for the current level. The default values are calculated by means of the *width* of a box by setting a *value* to the current counter using ‘0’ for `\arabic*`, ‘M’ for `\Alph*`, ‘m’ for `\alph*`, ‘VIII’ for `\Roman*` and ‘viii’ for `\roman*`.

`widest = {⟨integer | string⟩}` default: *empty*

Sets the `labelwidth` key pass the *integer* or converting the *string* of the form `\Alph`, `\alph`, `\Roman` or `\roman` to a *value* for the current counter defined by `label` key, then calculating the *width* by means of a box. For example `widest={XXIII}` or `widest={23}` are equivalent. This key is useful when the default values of the `labelwidth` key are smaller than those actually used.

`font = {⟨font commands⟩}` default: *empty*

Sets the *font style* for the current *label* defined by `label` key. For example `font={\bfseries\small}`.

`align = {⟨left | right | center⟩}` default: *left*

Sets the *aligned* of *label* defined by `label` key on the current level in the label box.

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

Wraps the current *label* defined by `label` key referenced by `{#1}`. The `{⟨code⟩}` must be passed between braces. This key does not modify the value set by the `labelwidth` key and is applied only on `\item` and `\item*`. When using it in the `\setenumext` command it is necessary to use the *double hash* ‘`{#1}`’. For example `wrap-label={\fbox{#1}}` or you can create a command:

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

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

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

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

3.2 Keys for spaces

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

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

3.2.1 Vertical spaces

`topsep = {⟨rubber length | rigid length⟩}`

default: *by levels*

Set the *vertical space* added to both the top and bottom of the list. Internally sets the value of `\topsep` for the current level. The default values for first level 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.

`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 values for first level 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.

`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 are 2.0pt plus 1.0pt minus 1.0pt, for third and fourth level are 1.0pt minus 1.0pt.

- The value of this parameter also affects the *inner levels* and the *keyans* environment. Caution should be taken with “blank lines” or `\par` command “before” each environment or nested level when formatting the source code of document. T_EX 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 values for first level are `4.0pt` plus `2.0pt` minus `1.0pt`, for the rest of the levels are `2.0pt` plus `1.0pt` minus `1.0pt`.

`noitemsep` *<value forbidden>* default: *not used*

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

`nosep` *<value forbidden>* default: *not used*

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

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

`above` = {*<rubber length | rigid length>*} default: *not used*

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

`above*` = {*<rubber length | rigid length>*} default: *not used*

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

`below` = {*<rubber length | rigid length>*} default: *not used*

Set the *extra vertical space* 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* 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*”.

3.2.2 Horizontal spaces

`itemindent` = {*<rigid length>*} default: `0pt`

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

`rightmargin` = {*<rigid length>*} default: `0pt`

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

`listparindent` = {*<rigid length>*} default: `0pt`

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

`list-offset` = {*<rigid length>*} default: `0pt`

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

`list-indent` = {*<rigid length>*} default: `labelwidth + labelsep`

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

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

3.3 Keys for add code

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

`before` = {*<code>*} default: *not used*

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

`before*` = {*<code>*} default: *not used*

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

`first = {\code}` default: *not used*
 Executes `{\code}` when “starting” the environment. The `{\code}` must be passed between braces, is executed right “after” all *list parameters* are done, after the second argument of `list`, just before the first occurrence of `\item`: `\list{\arg one}{\arg two}{\code}\item`.

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

`after = {\code}` default: *not used*
 Execute `{\code}` “after” finishing the environment. The `{\code}` must be passed between braces.

3.4 Keys for start and resume

`start = {\integer | string}` default: `1`
 Sets the *start value* of the numbering on the current level. Internally `\string` is passed as value to the counter defined by `label` key on the current level, i.e. it is equivalent to enter `start=5`, `start=E` or `start=v`.

`resume \value forbidden` default: *not used*
 Sets the *start* to value from the previous of the counter defined by `label` key for the “first level”. This `\key` does not receive an argument. The `\key` can be overwritten using the `start` key. If the `save-ans` key is present and `{\store name}` exist, the numbering will continue according to this key. This key is “only” available for the “first level” of `enumext`.

3.5 Keys for multicol

`columns = {\integer}` default: `1`
 Set the *number of columns* to be used by the `multicol` environment within the environment. The value must be a positive integer less than or equal to `10`.

`columns-sep = {\rigid length}` default: *by level*
 Set the *space between columns* used by the `multicol` environment within the environment. Internally sets the value of `\columnsep`, by default its value is equal to the sum of the values set in the keys `labelwidth` and `labelsep` of the current level.

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

3.6 Keys for minipage

`mini-env = {\rigid length}` default: *not used*
 Sets the *width* of the `minipage` environment on the “right side”. This value added to the value set by the `mini-sep` key to determines the *width* of the `minipage` environment on the “left side”, taking `\linewidth` as the maximum reference value.

`mini-sep = {\rigid length}` default: `0.3333em`
 Sets the *space between* the `minipage` environment on the “left side” and the `minipage` environment on the “right side”. This separation is applied together with `\hfill`.

3.6.1 The command `\miniright`

`\miniright` The `\miniright` command close the `minipage` environment on the “left side” and opens the `minipage`
`\miniright*` 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 version* ‘`*`’ inhibits the use of `\centering` command i.e. the usual L^AT_EX justification is maintained in the `minipage` on the “right side”.

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

4 The storage system

The entire mechanism for “storing content” it is activated according to `save-ans` key on the “first level” of `enumext` environment. Only when this `\key` is “active” the `\anskey` command and the environments `keyans` and `keyanspic` are available.

<pre>\begin{enumext}[save-ans={\store name}] \item Text \begin{keyans} ... \end{keyans} \end{enumext}</pre>	<pre>\begin{enumext}[save-ans={\store name}] \item Text \begin{keyanspic} ... \end{keyanspic} \end{enumext}</pre>
---	---

4.1 Keys for storage

- `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` in `enumext` environment, `\item*` in `keyans` environment and `\anspic*` in `keyanspic` environment. If the ⟨sequence⟩ or ⟨prop list⟩ does not exist, it will be created globally.
- `wrap-ans = {⟨code {#1} more code⟩}` default: `\fbox`
 Wraps the current ⟨argument⟩ passed `\anskey` command to referenced by {#1}. The {⟨code⟩} must be passed between braces. This ⟨key⟩ only affects the current ⟨argument⟩ passed to `\anskey` and NOT the “stored content” in the ⟨store name⟩ set by `save-ans` key. If this key is passed using the `\setenumext` command it is necessary to use double ‘{##1}’.
- `mark-ans = {⟨symbol⟩}` default: `\textasteriskcentered`
 Sets the *symbol* to be displayed in the left margin of the “stored content” in ⟨store name⟩ set by `save-ans` key when using `show-ans` key.
- `mark-pos = {⟨left | right⟩}` default: *left*
 Sets the aligned of the *symbol* defined by `mark-ans` key. The “symbol” is aligned in a box with the same dimensions of the label box defined by `labelwidth` key on the current level and separated by the value of the `labelsep` key.
- `show-ans = {⟨true | false⟩}` default: *false*
 Displays the current ⟨argument⟩ passed to `\anskey` in `enumext` environment, the current ⟨label⟩ for `\item*` in `keyans` environment and the current ⟨label⟩ for `\anspic*` in `keyanspic` environment at the place where it is executed. If the optional argument is present in `\item*` or `\anspic*` it will be shown in square brackets.
- `show-pos = {⟨true | false⟩}` default: *false*
 Displays the *position* occupied by the “stored content” by `\anskey` in `enumext` environment, `\item*` in `keyans` environment and `\anspic*` in `keyanspic` environment in ⟨store name⟩ set by `save-ans` key. This position is used by the `\getkeyans` command and by the `\ref` command if the `store-ref` key is active.

4.2 Keys for internal label and ref

- `store-ref = {⟨true | false⟩}` default: *false*
 Activates the internal “label and ref” mechanism for referencing “stored content” in ⟨store name⟩ set by `save-ans` key. To reference the location of the “stored content” within the environment you must use `\ref{⟨store name: position⟩}`, where ⟨position⟩ corresponds to the position occupied by the “stored content” in the ⟨store name⟩ returned by the `show-pos` key. For example `\ref{test:4}` will return 3. (b) which corresponds to the location of the “stored content” at position 4 within the environment in which the key `save-ans=test` was set.
- `mark-ref = {⟨symbol⟩}` default: `\textasteriskcentered`
 Sets the *symbol* that will be displayed by the `\printkeyans` command only if the `hyperref` package is detected and the `store-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.

4.3 Keys for check answers

- `check-ans = {⟨true | false⟩}` default: *false*
 Enables the “checking answer” mechanism. This key works under the logic that each question will contain “only one answer”, it is intended to be used in conjunction with `no-store` key.
- `no-store` ⟨value forbidden⟩ default: *not used*
 This is a “meta-key” that does not receive an argument. This key is used in conjunction with `check-ans` and is designed to be used with nested levels of `enumext` in which the `\anskey` command will not be used.

4.4 The command `\anskey`

`\anskey` `\anskey{⟨content⟩}`

The `\anskey` command takes a mandatory argument and is triggered by `save-ans` key. The “content” are “stored” in ⟨store name⟩ set by `save-ans` key. The command does “not support” verbatim content and must NOT be nested. By design it is assumed that each `\item` or `\item*` will have a “single” occurrence of the command unless a nested level is opened or the `no-store` key is used. If `store-ref` key are active and the `hyperref`[7] package is detected, `\hyperlink` and `\hypertarget` will be used, otherwise the usual “label and ref” system provided by L^AT_EX will be used.

Example

- | | |
|---|---|
| <ul style="list-style-type: none"> ★ 1. Text containing our instructions or questions. <li style="margin-left: 20px;">* first answer 2. Text containing our instructions or questions. <li style="margin-left: 20px;">(a) Question. <li style="margin-left: 40px;">* second answer | <ul style="list-style-type: none"> 3. Text containing our instructions or questions. <li style="margin-left: 20px;">* third answer 4. Text containing our instructions or questions. <li style="margin-left: 20px;">* fourth answer |
|---|---|

```
\begin{enumext}[save-ans=test,show-ans]
  \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}
```

4.5 The environment keyans

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

The `keyans` is an “*enumerated list*” environment designed for “*multiple choice*” questions activated by the `save-ans` key. This environment can NOT be nested and must always be at the “*first level*” of the `enumext` environment, the commands `\item` and `\item[\langle custom \rangle]` work in the usual.

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

The `\keys` set in the optional argument of the environment are the same (almost) as those of the `enumext` environment and have higher precedence than those set by `\setenumext[\langle keys \rangle]{\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*)`.

4.5.1 The `\item*` in `keyans`

```
\item* \item*
\item* [\langle content \rangle]
```

The `\item*` and `\item*[\langle content \rangle]` command store the current `\label` set by `label` key next to the `\content` (if it is present) in `\store name` set by `save-ans` key in the “*first level*” of the `enumext` environment.

The *starred version* ‘`*`’ 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 *starred version* ‘`*`’ will only appear “*once*” within the environment.

🔗 The behavior of `\item*` in `keyans` environment is NOT the same as in the `enumext` environment.

Example

```
\begin{enumext}[save-ans=test,columns=2,show-ans]
  \item Text containing a question.
    \begin{keyans}[nosep]
      \item Choice
      \item* Correct choice
      \item Choice
      \item Choice
    \end{keyans}

  \item Text containing a question and image.
    \begin{keyans}[nosep,mini-env={0.4\linewidth}]
      \item Choice
      \item Choice
      \item Choice
      \item Choice
      \item*[\note] Correct choice
      \miniright
      \includegraphics[scale=0.25]{example-image-a}

      Some text
    \end{keyans}
\end{enumext}
```

1. Text containing a question.

(A) Choice

* (B) Correct choice

(C) Choice

(D) Choice
2. Text containing a question and image.

(A) Choice

(B) Choice

(C) Choice

(D) Choice

* (E) [note] Correct choice



4.6 The environment keyanspic

keyanspic

`\begin{keyanspic}[\langle number above, number below \rangle]\anspic{\langle drawing \rangle}\anspic*[\langle content \rangle]{\langle drawing \rangle}`

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

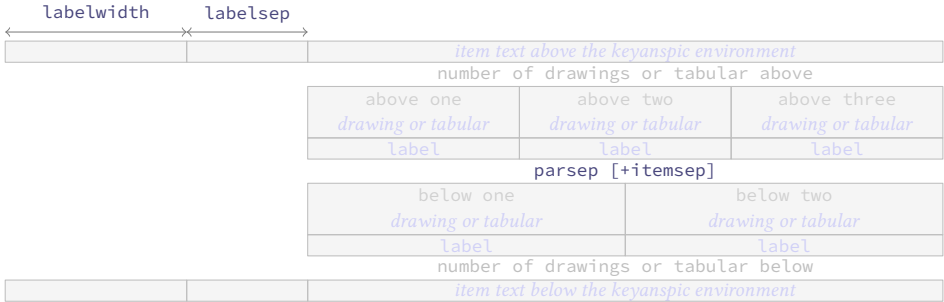


Figure 6: Representation of the `keyanspic` environment with optional argument `[3,2]` in `enumext`.

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

4.6.1 The command \anspic

\anspic

`\anspic{\langle drawing or tabular \rangle}`
`\anspic*[\langle content \rangle]{\langle drawing or tabular \rangle}`

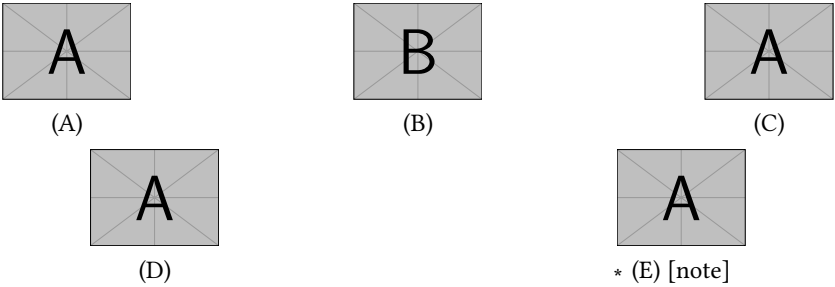
The `\anspic` command take three arguments, the *starred version* “*” store the current `\label` next to the `\content` (if it is present) in `\store name` set by `save-ans` key.

The *starred version* “*” 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 version* “*” will only appear “once” within the environment.

Example

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

1. Question with images.



4.7 Printing stored content

4.7.1 The command \getkeyans

`\getkeyans` `\getkeyans{<store name> : <position>}`

The command `\getkeyans` prints the “only stored content” in `<store name>` defined by `save-ans` key in the `<position>` returned by the `show-pos` key.

The “content” can only be accessed “after” it is stored, if the `<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 `store-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.

4.7.2 The command \printkeyans

`\printkeyans` `\printkeyans[<keys>]{<store name>}`

The command `\printkeyans` prints “all stored content” in `{<store name>}` defined by `save-ans` key. The “content” can only be accessed “after” it is stored, if `<store name>` does not exist the command will return an error.

Internally it places the “stored content” inside the `enumext` environment with default values for `label` key are the same as those of the `enumext` environment along with the keys: `nosep`, `first=\small`, `font=\small` for all levels, except for the first one that adds the `columns=2` key.

The optional argument allows to handle the `<keys>` “on the first level” of the `enumext` environment encapsulated by the command. If need to pass options for nested levels use `\setenumext[<print> , <level>]{<store name>}`.

Example

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

  \begin{enumext}[nosep]
    \item \LaTeXe 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) ~~LaTeXe~~ 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
- *

*

*

*

*

5 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: $1 \text{ \AA} = 1 \times 10^{-10}$ 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

 $1 \text{ \AA} = 1 \times 10^5 \text{ fm}$.

B

 $1 \text{ \AA} = 1 \times 10^{-5} \text{ fm}$.

C

 $1 \text{ \AA} = 1 \times 10^{-15} \text{ fm}$.

D

 $1 \text{ \AA} = 1 \times 10^3 \text{ fm}$.
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: $1 \text{ \AA} = 1 \times 10^{-10}$ 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

 $1 \text{ \AA} = 1 \times 10^5 \text{ fm}$.

B

 $1 \text{ \AA} = 1 \times 10^{-5} \text{ fm}$.


C

 $1 \text{ \AA} = 1 \times 10^{-15} \text{ fm}$.

D

 $1 \text{ \AA} = 1 \times 10^3 \text{ fm}$.
1. B
2. A
3. B
4. A

Example 2

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

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

A

 36 km/h.

✓

 B

 360 km/h.

C

 27,8 km/h.

D

 $3,60 \times 10^8$ km/h.
2. In fisica nucleare si usa l'angstrom (simbolo: $1 \text{ \AA} = 1 \times 10^{-10}$ 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

 $1 \text{ \AA} = 1 \times 10^5 \text{ fm}$.

B

 $1 \text{ \AA} = 1 \times 10^{-5} \text{ fm}$.

C

 $1 \text{ \AA} = 1 \times 10^{-15} \text{ fm}$.

D

 $1 \text{ \AA} = 1 \times 10^3 \text{ fm}$.
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: $1 \text{ \AA} = 1 \times 10^{-10}$ 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

 $1 \text{ \AA} = 1 \times 10^5 \text{ fm}$.

B

 $1 \text{ \AA} = 1 \times 10^{-5} \text{ fm}$.

C

 $1 \text{ \AA} = 1 \times 10^{-15} \text{ fm}$.

D

 $1 \text{ \AA} = 1 \times 10^3 \text{ fm}$.
1. B
2. A
3. B
4. A
- *
- *
- *
- *

¹The cool TeX automation tool: <https://www.ctan.org/pkg/arara>

©2024 by Pablo González L

14 / 104

Example 3

A “simple multiple choice” test 📄.

1. First type of questions
- A

 value

B

 correct

C

 value

D

 value
2. Second type of questions
- I. $2\alpha + 2\delta = 90^\circ$

II. $\alpha = \delta$

III. $\angle EDF = 45^\circ$

A

 I only

B

 II only

C

 I and II only

D

 I and III only

E

 I, II, and III
3. Third type of questions
- (1) $2\alpha + 2\delta = 90^\circ$

(2) $\angle EDF = 45^\circ$

A

 value

B

 value

C

 value

D

 value

E

 value
4. Question with image and label below:



A



B



C



D



E

5. Question with image on left side:

- A

 value
- B

 value
- C

 value
- D

 correct
- E

 value



Test keys

1. B $x = 5$
2. D
3. C some note
4. B
5. D other note

Example 4

A “simple worksheet” using ducks :) 📄.

- 1

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

 Factor $3x + 3y + 3z$
- The following questions need to be cuaqtified :)
- 3

 True False
- (a)

 $\alpha > \delta$
- (b)

~~ETX~~ze is cool?
- 4

 Related to Linux
- (a)

 You use linux?
- (b)

 Usually uses the package manager?
- (c)

 Rate the following package and class
- i.

 xsim-exam
- ii.

 xsim
- iii.

 exsheets

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

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

 Yes, dnf
- (c)

 i. doesn't exist for now :(
- ii. very good
- iii. obsolete

Example 5

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

<div>1</div> <p>Which choice best describes what happens in the passage?</p> <p>A) One character argues with another character who intrudes on her home.</p> <p>B) One character receives a surprising request from another character.</p> <p>C) One character reminisces about choices she has made over the years.</p> <p>D) One character criticizes another character for pursuing an unexpected course of action.</p>	<div>3</div> <p>Which choice best describes what happens in the passage?</p> <p>A) One character argues with another character who intrudes on her home.</p> <p>B) One character receives a surprising request from another character.</p> <p>C) One character reminisces about choices she has made over the years.</p> <p>D) One character criticizes another character for pursuing an unexpected course of action.</p>
<div>2</div> <p>Which choice best describes what happens in the passage?</p> <p>A) One character argues with another character who intrudes on her home.</p> <p>B) One character receives a surprising request from another character.</p> <p>C) One character reminisces about choices she has made over the years.</p> <p>D) One character criticizes another character for pursuing an unexpected course of action.</p>	<div>4</div> <p>Which choice best describes what happens in the passage?</p> <p>A) One character argues with another character who intrudes on her home.</p> <p>B) One character receives a surprising request from another character.</p> <p>C) One character reminisces about choices she has made over the years.</p> <p>D) One character criticizes another character for pursuing an unexpected course of action.</p>

1. A)

2. C)

3. B)

4. D)

6 The way of non-enumerated lists

It is possible to use (or abuse) the enumext environment to mimic non-enumerated list environments such as itemize and description, clearly the <keys> to “store answers”, the keyans and keyanspic environments lose their sense and it is not the focus of the main of this package, but, why not to do it?. Here I leave as an example other uses of the enumext environment that can be helpful for specific purposes. The “trick” to generate these fake environments is set label={} or label={<some>} and play with the list-indent, list-offset, font and wrap-label keys.

Fake itemize environment

Here we set the label key using the default settings in L^AT_EX for the four levels \textbullet, \textendash, \textasteriskcentered and \textperiodcentered together with the nosepe 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 nosepe key

- First level item
 - Second level item
 - * Third level item
 - Fourth level item
 - First level item
- * First level item
 - ◇ Second level item
 - Third level item
 - ★ Fourth level item
 - * First level item

Fake description environment

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

- Something** A short one-line description.

This is an entry *without* a label.

Something A short *one-line* description text.

Something long A much *longer* description text may take more than one line or more than one paragraph.

 Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

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

- Something** A short one-line description.

This is an entry *without* a label.

Something A short *one-line* description text.

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

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

Description indented by label

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

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

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

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

Something A short one-line description.

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

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

Description with multi-line labels

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

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

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

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

Something A short one-line description.

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

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

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

LoNg 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[4]` 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`[10] and `l3seq`[10] modules together with the `hyperref`[7] and `enumitem`[5] packages, which did the job, but with some limitations.

As time went by I took these limitations as a personal challenge which I called “*reinventing the wheel*”, since there were packages and classes that did more or less what I was looking for, but did not fit my simple requirements. This “*reinventing the wheel*” finally ended up becoming `enumext`.

Why list environments?

The answer is simple, first I love the beauty of its syntax and many of what I had already written used the `enumerate` environment or lists created using the `enumitem` package. In my mind I thought: how complicated could it be to write a package that looked like `enumitem`? It seemed simple enough, of course I didn’t have in mind the mess I was getting into working with `list` environments, `minipage` and adding support for the `multicol` and `hyperref` packages.

Of course, seeing the final result of the experiment “*reinventing the wheel*” I am quite satisfied.

Why not random questions and other utilities

The “*random*” type questions I love and hate them at the same time, although they simplify a lot the work when creating a multiple choice test, but you lose the beauty of typesetting a document with \LaTeX , that is to say the output does not always look as nice as it should, even if they are only alternatives these must follow a certain order when presented either numerical or presentation, that said handling that using *nested lists* is quite complicated so I do not classify to be implemented.

7 References

- [1] HIRSCHHORN, PHILIP. “Using the exam document class”. Available from CTAN, <https://www.ctan.org/pkg/exam>, 2023.
- [2] NIEDERBERGER, CLEMENS. “xsim – eXercise Sheets IMproved”. Available from CTAN, <https://www.ctan.org/pkg/xsim>, 2023.
- [3] MITTELBACH, FRANK. “An environment for multicolumn output”. Available from CTAN, <https://www.ctan.org/pkg/multicol>, 2024.
- [4] The \LaTeX Project. “enumerate – Enumerate with redefinable labels”. Available from CTAN, <https://www.ctan.org/pkg/enumerate>, 2024.
- [5] BEZOS, JAVIER. “Customizing lists with the enumitem package”. Available from CTAN, <https://www.ctan.org/pkg/enumitem>, 2019.
- [6] BERRY, KARL. “ \LaTeX 2_ε: An Unofficial Reference Manual”. Available from CTAN, <https://ctan.org/pkg/latex2e-help-texinfo>, 2024.
- [7] The \LaTeX Project. “Extensive support for hypertext in \LaTeX ”. Available from CTAN, <https://www.ctan.org/pkg/hyperref>, 2024.
- [8] BURNOL, JEAN-FRANÇOIS. “The footnotehyper package”. Available from CTAN, <https://www.ctan.org/pkg/footnotehyper>, 2021.
- [9] The \LaTeX Project. “The expl3 package”. Available from CTAN, <https://www.ctan.org/pkg/l3kernel>, 2024.
- [10] The \LaTeX Project. “The \LaTeX 3 Interfaces”. Available from CTAN, <https://www.ctan.org/pkg/l3kernel>, 2024.
- [11] The \LaTeX Project. “The xparse package”. Available from CTAN, <https://www.ctan.org/pkg/xparse>, 2024.
- [12] GUNDLACH, PATRICK. “The lua-visual-debug package”. Available from CTAN, <https://www.ctan.org/pkg/lua-visual-debug>, 2023.
- [13] LEMVIG, MOGENS. “The shortlst package”. Available from CTAN, <https://www.ctan.org/pkg/shortlst>, 1998.
- [14] NIEDERBERGER, CLEMENS. “tasks – Horizontally columned lists”. Available from CTAN, <https://www.ctan.org/pkg/tasks>, 2022.

8 Change history

v1.0 2024-04-22 – First public release.

9 Index of Documentation

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

C

Document class:

article 1

book 1

exam 2

letter 1

report 1

\columnbreak 4

\columnsep 9

Commands provide by enumext:

\anskey 3, 9–11

\anspic* 3, 10, 12

\anspic 12

\getkeyans 3, 10, 13

\item* 3–6, 10, 11

\item 5, 6, 9–11

\miniright 3, 4, 9

\printkeyans 3, 5, 10, 13

\setenumext 3, 5, 6, 10, 11, 13

Counters defined by enumext:

enumXiii 3

enumXii 3

enumXiv 3

enumXi 3

enumXviii 3

enumXvii 3

enumXvi 3

enumXv 3

E

Environments provide by enumext:

enumext* 3, 4

enumext 3–5, 9–11, 13, 16

keyans* 3, 4

keyanspic 3, 6, 9, 10, 12, 16

keyans 3–7, 9–12, 16

Environments:

enumerate 1–3, 5, 18

list 3, 8, 18

minipage 2–4, 9, 18

multicols 2, 4, 9

I

\item 3, 4

\itemsep 8

K

Keys for environments provide by enumext:

above* 8

above 8

after 9

align 6, 17

before* 8

before 8

below* 8

below 8

check-ans 10

columns-sep 4, 9

columns 4, 8, 9

first 9

font 6

item-pos* 5

item-sym* 5

itemindent 8

itemsep 8, 12

labelsep 3, 5, 6, 8–10, 17

labelwidth 3, 5, 6, 8–10, 17

label 6, 9, 11, 13, 16, 17

list-indent 3, 8

list-offset 3, 8, 17

listparindent 8

mark-ans 10

mark-pos 10

mark-ref 10

mini-env 4, 8, 9

mini-sep 4, 9

no-store 10

noitemsep 8

nosep 8, 16

parsep 7, 8, 12

partopsep 7

ref 4, 6

resume 9

rightmargin 8

save-ans 4, 9–13

show-ans 10

show-length 6

show-pos 10, 13

start 9

store-ref 4, 6, 10, 13

topsep 7, 8

widest 6

wrap-ans 10

wrap-label* 6, 17

wrap-label 6

L

\label 4

Labels provide by enumext:

\Alph* 6, 11

\Roman* 6

\alph* 6

\arabic* 6

\roman* 6

\labelsep 3, 6

\labelwidth 3, 6

\linewidth 9

\listparindent 8

P

Packages:

enumerate 17

enumext 1–3, 12, 17, 18

enumitem 3, 4, 8, 17, 18

footnotehyper 4

hyperref 4, 10, 18

l3prop 1, 18

l3seq 1, 18

multicol 1, 4, 18

xsim 2

\parsep 7

\partopsep 7

©2024 by Pablo González L

19 / 104

R		\rightmargin	8
\raggedcolumns	4		
		T	
\ref	4	\topsep	7

10 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>.

10.1 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.

10.2 Initial set up

Start the DocStrip guards.

```
1 <*package>
```

Identify the internal prefix (L^AT_EX3 DocStrip convention) for l3doc class.

```
2 <@@=enumext>
```

10.3 Declaration of the package

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

```
3 \NeedsTeXFormat{LaTeX2e}[2023-11-01]
```

Finally we declare the `enumext` package.

```
4 \ProvidesExplPackage
5   {enumext}
6   {2024-04-22}
7   {1.0}
8   {Enumerate exercise sheets}
```

Then check if the `multicol` package is loaded, if not we load it.

```
9 \hook_gput_code:nnn {begindocument} {enumext}
10 {
11   \IfPackageLoadedTF { multicol }
12   {
13     \msg_info:nnn { enumext } { package-load } { multicol }
14   }
15   {
16     \msg_info:nnn { enumext } { package-not-load } { multicol }
17     \RequirePackage{multicol}[2023-03-30]
18   }
19 }
```

10.4 Definition of variables

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

Integer variables will control the nesting levels of the environments and boolean variables will be used to determine if they are present (nested) in each other.

```
20 \int_new:N \__enumext_level_int
21 \int_new:N \__enumext_level_h_int
22 \int_new:N \__enumext_keyans_level_int
23 \int_new:N \__enumext_keyans_level_h_int
24 \int_new:N \__enumext_keyans_pic_level_int
25 \bool_new:N \__enumext_starred_bool
26 \bool_new:N \__enumext_standar_bool
27 \bool_new:N \__enumext_keyans_env_bool
```

(End of definition for `__enumext_level_int` and others.)

```

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

```

Variables to store the “*name of the counters*” `enumXi`, `enumXii`, `enumXiii` and `enumXiv` for `enumext` environment, `enumXv` for `keyans` environment and `enumXvi` for the `keyanspic` environment.

The counters `enumXvii` and `enumXviii` are used by `enumext*` and `keyans*` environments.

The initial values of these variables are set by the function `__enumext_define_counters:Nn` and then modified by the function `__enumext_label_style:Nnn` used by `label` key (§10.8).

```

30 \cs_set_protected:Npn \__enumext_tmp:n #1
31 {
32   \tl_new:c { l__enumext_counter_#1_tl }
33 }
34 \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_resume_bool
\g__enumext_resume_int
\l__enumext_resume_vii_bool
\g__enumext_resume_vii_int
\g__enumext_item_symbol_tl

```

The boolean variable `\l__enumext_resume_bool` is used by `resume` key, the value from which the environment’s will start is stored in the integer variable `\g__enumext_resume_int` (§10.21). The global token list `\g__enumext_item_symbol_tl` is used by `item-sym*` key (§10.26).

```

35 \bool_new:N \l__enumext_resume_bool
36 \int_new:N \g__enumext_resume_int
37 \bool_new:N \l__enumext_resume_vii_bool
38 \int_new:N \g__enumext_resume_vii_int
39 \tl_new:N \g__enumext_item_symbol_tl

```

(End of definition for `\l__enumext_resume_bool` 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` (§10.12) and `label` (§10.10) keys.

```

40 \dim_new:N \l__enumext_current_widest_dim
41 \tl_new:N \g__enumext_counter_styles_tl
42 \tl_new:N \g__enumext_widest_label_tl
43 \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 (§10.14).

The variables `\l__enumext_leftmargin_X_dim` and `\l__enumext_itemindent_X_dim` are used (and set) by the function `__enumext_calc_hspace:NNNNNNNNNN` (§10.30) which determines the internal values for `\leftmargin` and `\itemindent`.

```

44 \cs_set_protected:Npn \__enumext_tmp:n #1
45 {
46   \bool_new:c { l__enumext_leftmargin_tmp_#1_bool }
47   \dim_new:c { l__enumext_leftmargin_tmp_#1_dim }
48   \dim_new:c { l__enumext_leftmargin_#1_dim }
49   \dim_new:c { l__enumext_itemindent_#1_dim }
50 }
51 \clist_map_inline:nn { i, ii, iii, iv, v, vi, vii, viii } { \__enumext_tmp:n {#1} }

```

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

```

\l__enumext_multicols_above_X_skip
\l__enumext_multicols_below_X_skip

```

Internal variables used by `columns` key §10.18).

```

52 \cs_set_protected:Npn \__enumext_tmp:n #1
53 {
54   \skip_new:c { l__enumext_multicols_above_#1_skip }
55   \skip_new:c { l__enumext_multicols_below_#1_skip }
56 }
57 \clist_map_inline:nn { i, ii, iii, iv, v } { \__enumext_tmp:n {#1} }

```

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

```

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

```

Internal variables used by `\miniright` command (§10.19.4) and the keys `miniright`, `miniright*`, `mini-env` and `mini-sep` (§10.17, §10.19).

```

58 \int_new:N \g__enumext_minipage_stat_int
59 \skip_new:N \l__enumext_minipage_left_skip
60 \skip_new:N \l__enumext_minipage_right_skip
61 \skip_new:N \l__enumext_minipage_after_skip
62 \skip_new:N \g__enumext_minipage_right_skip
63 \skip_new:N \g__enumext_minipage_after_skip
64 \cs_set_protected:Npn \__enumext_tmp:n #1

```

```

65   {
66     \dim_new:c { \__enumext_minipage_left_#1_dim }
67     \bool_new:c { \__enumext_minipage_active_#1_bool }
68   }
69   \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.)

```

\__enumext_wrap_label_X_bool
\__enumext_wrap_label_opt_X_bool
\__enumext_start_X_int
\__enumext_fake_item_indent_X_tl
\__enumext_label_fill_left_X_tl
\__enumext_label_fill_right_X_tl
\__enumext_vspace_a_star_X_bool
\__enumext_vspace_b_star_X_bool

```

The integer variable `__enumext_start_X_int` are used by the `start` key (§10.12), the token list `__enumext_fake_item_indent_X_tl` is used by `itemindent` key, the variables `__enumext_label_fill_left_X_tl` and `__enumext_label_fill_right_X_tl` are used by the `align` key (§10.10). The boolean vars `__enumext_vspace_a_star_X_bool`, `__enumext_vspace_b_star_X_bool` are used by `above`, `above*`, `below` and `below*` keys

```

70   \cs_set_protected:Npn \__enumext_tmp:n #1
71   {
72     \bool_new:c { \__enumext_wrap_label_#1_bool }
73     \bool_new:c { \__enumext_wrap_label_opt_#1_bool }
74     \int_new:c { \__enumext_start_#1_int }
75     \tl_new:c { \__enumext_fake_item_indent_#1_tl }
76     \tl_new:c { \__enumext_label_fill_left_#1_tl }
77     \tl_new:c { \__enumext_label_fill_right_#1_tl }
78     \bool_new:c { \__enumext_vspace_a_star_#1_bool }
79     \bool_new:c { \__enumext_vspace_b_star_#1_bool }
80   }
81   \clist_map_inline:nn { i, ii, iii, iv, v, vii, viii } { \__enumext_tmp:n {#1} }

```

(End of definition for `__enumext_wrap_label_X_bool` and others.)

```

\__enumext_store_active_bool
\__enumext_store_name_tl
\g__enumext_store_name_tl
\__enumext_store_anskey_arg_tl
\__enumext_store_columns_join_int
\__enumext_store_keyans_label_tl
\__enumext_keyans_tmpa_tl

```

The boolean variable `__enumext_store_active_bool` setting by `save-ans` key (§10.21) activates all the mechanism related to `\anskey`, `keyans`, `keyans*` and `keyanspic`.

The variable `__enumext_store_name_tl` sets the name for the storage in *⟨sequence⟩* and *⟨prop list⟩*, the variable `\g__enumext_store_name_tl` is just a copy of the storage name used by the `check-ans` key (§10.21).

The variable `__enumext_store_anskey_arg_tl` stores the contents of `\anskey` (§10.24) and the variable `__enumext_store_keyans_label_tl` stores the contents of `\item*` (§10.28.2) for the `keyans` and `keyans*` environments and the contents of `\anspic*` (§10.34.1) for the `keyanspic` environment.

The variable `__enumext_keyans_tmpa_tl` is a temporary variable used by `keyans` and `keyanspic` at various points.

```

82   \bool_new:N \__enumext_store_active_bool
83   \tl_new:N \__enumext_store_name_tl
84   \tl_new:N \g__enumext_store_name_tl
85   \tl_new:N \__enumext_store_anskey_arg_tl
86   \int_new:N \__enumext_store_columns_join_int
87   \tl_new:N \__enumext_store_keyans_label_tl
88   \tl_new:N \__enumext_keyans_tmpa_tl

```

(End of definition for `__enumext_store_active_bool` and others.)

```

\__enumext_setkey_tmpa_tl
\__enumext_setkey_tmpp_tl
\__enumext_setkey_tmpa_int
\__enumext_setkey_tmpa_seq
\__enumext_setkey_tmpp_seq

```

Internal variables used by the command `\setenumext` (§10.38).

```

89   \tl_new:N \__enumext_setkey_tmpa_tl
90   \tl_new:N \__enumext_setkey_tmpp_tl
91   \int_new:N \__enumext_setkey_tmpa_int
92   \seq_new:N \__enumext_setkey_tmpa_seq
93   \seq_new:N \__enumext_setkey_tmpp_seq

```

(End of definition for `__enumext_setkey_tmpa_tl` and others.)

```

\__enumext_store_opt_X_tl
\__enumext_print_keyans_X_tl
\__enumext_store_columns_X_bool
\__enumext_store_columns_X_int
\__enumext_store_columns_sep_X_bool
\__enumext_store_columns_sep_X_dim
\__enumext_store_upper_level_X_bool

```

Internal variables used by `[⟨key = val⟩]` in `enumext` and `enumext*` environment, the command `\printkeyans` (§10.37) and the keys `columns*` and `columns-sep*`.

```

94   \cs_set_protected:Npn \__enumext_tmp:n #1
95   {
96     \tl_new:c { \__enumext_store_opt_#1_tl }
97     \tl_new:c { \__enumext_print_keyans_#1_tl }
98     \bool_new:c { \__enumext_store_columns_#1_bool }
99     \int_new:c { \__enumext_store_columns_#1_int }
100    \bool_new:c { \__enumext_store_columns_sep_#1_bool }
101    \dim_new:c { \__enumext_store_columns_sep_#1_dim }
102    \bool_new:c { \__enumext_store_upper_level_#1_bool }
103  }
104   \clist_map_inline:nn { i, ii, iii, iv, vii } { \__enumext_tmp:n {#1} }

```

(End of definition for `\l__enumext_store_opt_X_tl` and others.)

```
\l__enumext_show_answer_bool
\l__enumext_show_position_bool
\l__enumext_mark_ref_sym_tl
\l__enumext_mark_answer_sym_tl
\l__enumext_mark_position_str
```

Internal variables for “*storage system*” mechanism used by `\anskey` (§10.24), `keyans` and `keyanspic` environments. These variables are used by `show-ans`, `show-pos`, `mark-ans`, `save-key` and `mark-ref` keys (§10.23).

```
105 \bool_new:N \l__enumext_show_answer_bool
106 \bool_new:N \l__enumext_show_position_bool
107 \tl_new:N \l__enumext_mark_ref_sym_tl
108 \tl_new:N \l__enumext_mark_answer_sym_tl
109 \str_new:N \l__enumext_mark_position_str
```

(End of definition for `\l__enumext_show_answer_bool` and others.)

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

Internal variables used by `keyanspic` environment (§10.34.2).

```
110 \seq_new:N \l__enumext_keyans_pic_body_seq
111 \dim_new:N \l__enumext_keyans_pic_width_dim
112 \int_new:N \l__enumext_keyans_pic_above_int
113 \int_new:N \l__enumext_keyans_pic_below_int
114 \skip_new:N \l__enumext_keyans_pic_above_skip
```

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

```
\l__enumext_check_ans_bool
\g__enumext_check_ans_show_bool
\g__enumext_check_ans_show_h_bool
\g__enumext_check_ans_item_tl
\l__enumext_compare_items_ans_int
\g__enumext_count_item_ans_int
\g__enumext_count_item_all_int
\g__enumext_count_level_X_int
\g__enumext_count_item_X_int
```

Internal variables used by “*check answer*” mechanism (§10.22.1) controlled by the `check-ans` and `no-store` keys.

```
115 \bool_new:N \l__enumext_check_ans_bool
116 \bool_new:N \g__enumext_check_ans_show_bool
117 \bool_new:N \g__enumext_check_ans_show_h_bool
118 \tl_new:N \g__enumext_check_ans_item_tl
119 \int_new:N \l__enumext_compare_items_ans_int
120 \int_new:N \g__enumext_count_item_ans_int
121 \int_new:N \g__enumext_count_item_all_int
122 \cs_set_protected:Npn \__enumext_tmp:n #1
123 {
124   \int_new:c { g__enumext_count_level_#1_int }
125   \int_new:c { g__enumext_count_item_#1_int }
126 }
127 \clist_map_inline:nn { i, ii, iii, iv, vii } { \__enumext_tmp:n {#1} }
```

(End of definition for `\l__enumext_check_ans_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 (§10.7). The boolean variable `\l__enumext_footnotes_key_bool` determine if `hyperref` is load with key `hyperfootnotes=true`.

```
128 \bool_new:N \l__enumext_hyperref_bool
129 \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_store_write_aux_file_tl
\l__enumext_label_copy_X_tl
```

Internal variables are used when executing the `store-ref` key. 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` and the variable `\l__enumext_store_write_aux_file_tl` will be in charge of executing the writing code in the `.aux` file.

```
130 \tl_new:N \l__enumext_newlabel_arg_one_tl
131 \tl_new:N \l__enumext_newlabel_arg_two_tl
132 \tl_new:N \l__enumext_store_write_aux_file_tl
133 \cs_set_protected:Npn \__enumext_tmp:n #1
134 {
135   \tl_new:c { l__enumext_label_copy_#1_tl }
136 }
137 \clist_map_inline:nn { i, ii, iii, iv, v, vi, vii, viii } { \__enumext_tmp:n {#1} }
```

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

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

Internal variables used for redefinition of `\footnote`.

```
138 \int_new:N \g__enumext_footnote_int
139 \seq_new:N \g__enumext_footnote_arg_seq
140 \seq_new:N \g__enumext_footnote_int_seq
```

(End of definition for `\g__enumext_footnote_int`, `\g__enumext_footnote_arg_seq`, and `\g__enumext_footnote_int_seq`.)

Internal variables used by `ref` key (§10.17, §10.18).

```

141 \tl_const:Nn \c__enumext_counter_style_tl
142   { { arabic } { roman } { Roman } { alph } { Alph } }
143 \tl_new:N \l__enumext_ref_key_arg_tl
144 \tl_new:N \l__enumext_ref_aux_tl
145 \cs_set_protected:Npn \__enumext_tmp:n #1
146   {
147     \tl_new:c { l__enumext_counter_style_for_ref_#1_tl }
148     \tl_new:c { l__enumext_the_counter_#1_tl }
149     \tl_set:ce { l__enumext_the_counter_#1_tl } { \exp_not:c { theenumX#1 } }
150   }
151 \clist_map_inline:nn { i, ii, iii, iv, v, vi, vii, viii } { \__enumext_tmp:n {#1} }

```

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

Internal variables used by `enumext*` and `keyans*` environments.

```

152 \cs_set_protected:Npn \__enumext_tmp:n #1
153   {
154     \bool_new:c { l__enumext_item_starred_#1_bool }
155     \int_new:c { l__enumext_item_column_pos_#1_int }
156     \int_new:c { g__enumext_item_count_all_#1_int }
157     \int_new:c { l__enumext_joined_item_#1_int }
158     \int_new:c { l__enumext_joined_item_aux_#1_int }
159     \int_new:c { l__enumext_tmpa_#1_int }
160     \box_new:c { l__enumext_item_text_#1_box }
161     \dim_new:c { l__enumext_joined_width_#1_dim }
162     \dim_new:c { l__enumext_item_width_#1_dim }
163     \tl_new:c { g__enumext_item_symbol_aux_#1_tl }
164     \str_new:c { l__enumext_align_label_#1_str }
165     \bool_new:c { g__enumext_minipage_active_#1_bool }
166     \tl_new:c { g__enumext_miniright_code_#1_tl }
167     \bool_new:c { g__enumext_minipage_center_#1_bool }
168     \dim_new:c { g__enumext_minipage_right_#1_dim }
169     \skip_new:c { g__enumext_minipage_right_#1_skip }
170   }
171 \clist_map_inline:nn { vii, viii } { \__enumext_tmp:n {#1} }

```

(End of definition for `\l__enumext_item_starred_X_bool` and others.)

An internal `clist-var` variable to run with `__enumext_tmp:n`.

```

172 \tl_const:Nn \c__enumext_all_envs_clist
173   {
174     {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv},
175     {keyans}{v}, {enumext*}{vii}, {keyans*}{viii}
176   }

```

(End of definition for `\c__enumext_all_envs_clist`.)

10.5 Some utility functions

A internal “hook” function used for copying plain `list` and `minipage` environments definition and `hyperref` detection.

```

177 \cs_new_protected:Npn \__enumext_at_begin_document:n #1
178   {
179     \hook_gput_code:nnn {begindocument} {enumext} { #1 }
180   }

```

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

A internal “hook” function for execute code `minirigth` and `minirigth*` keys outside the `enumext*` and `keyans*` environments and print check-ans outside the `enumext` and `enumext*` environments.

```

181 \cs_new_protected:Npn \__enumext_after_env:nn #1 #2
182   {
183     \hook_gput_code:nnn {env/#1/after} {enumext} {#2}
184   }

```

(End of definition for `__enumext_after_env:nn`.)

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

```
185 \cs_new:Nn \__enumext_level:
186 {
187   \int_to_roman:n { \__enumext_level_int }
188 }
```

(End of definition for `__enumext_level:`.)

`__enumext_level_set:n` Function for set level in `enumext*`, `keyans*` and `keyans`.

```
\__enumext_level_end:n
189 \cs_new:Npn \__enumext_level_set:n #1
190 {
191   \cs_set_eq:cN { \__enumext_level_#1: } \__enumext_level:
192   \cs_set:Nn \__enumext_level: { #1 }
193 }
194 \cs_new:Npn \__enumext_level_end:n #1
195 {
196   \cs_set_eq:Nc \__enumext_level: { __enumext_level_#1: }
197 }
```

(End of definition for `__enumext_level_set:n` and `__enumext_level_end:n`.)

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

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

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

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

```
204 \cs_new:Npn \__enumext_show_length:nnn #1 #2 #3
205 {
206   * ~ #2
207   \prg_replicate:nn { 14 - \str_count:n {#2} } { ~ }
208   = ~ \use:c { #1_use:c } { \__enumext_#2_#3_#1 } \\
209 }
```

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

10.6 Copying list and minipage environments

The `list` environment provided by L^AT_EX has the following plain form:

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

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

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

```
210 \__enumext_at_begin_document:n
211 {
212   \cs_new_eq:NN \__enumext_start_list:nn \list
213   \cs_new_eq:NN \__enumext_stop_list: \endlist
214   \cs_new_eq:NN \__enumext_item_std:w \item
215 }
```

(End of definition for `__enumext_start_list:nn`, `__enumext_stop_list:`, and `__enumext_item_std:w`.)

The `minipage` environment provided by L^AT_EX has the following (simplified) plain form:

```
\minipage[⟨pos⟩][⟨height⟩][⟨inner-pos⟩]{⟨width⟩}
⟨internal implement⟩
\endminipage
```

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


```

\__enumext_minipage:w
\__enumext_endminipage:

```

The functions `__enumext_minipage:w`, `__enumext_endminipage:` and correspond to copies of `\minipage`, `\endminipage` from plain definition of `minipage` environment.

```

216 \__enumext_at_begin_document:n
217 {
218     \cs_new_eq:NN \__enumext_minipage:w \minipage
219     \cs_new_eq:NN \__enumext_endminipage: \endminipage
220 }

```

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

10.7 Compatibility with hyperref and footnotehyper

First we define the necessary rules using “hooks” to determine if the `hyperref` package is loaded.

```

221 \hook_gput_code:nnn { begindocument } { enumext } { \__enumext_after_hyperref: }
222 \hook_gset_rule:nnnn { begindocument } { enumext } { after } { hyperref }

```

```

\__enumext_after_hyperref:
\__enumext_hypertarget:nn
\__enumext_phantomsection:

```

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

```

223 \cs_new_protected:Nn \__enumext_after_hyperref:
224 {
225     \IfPackageLoadedTF { hyperref }
226     {
227         \msg_info:nnn { enumext } { package-load } { hyperref }
228         \bool_set_true:N \l__enumext_hyperref_bool
229         \IfHyperBoolean{hyperfootnotes}
230         {
231             \typeout{hyperfootnotes=true}
232             \bool_set_true:N \l__enumext_footnotes_key_bool
233         }
234         { \typeout{hyperfootnotes=false} }
235     }
236     { }

```

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`.

```

237 \bool_if:NT \l__enumext_footnotes_key_bool
238 {
239     \IfPackageLoadedTF { footnotehyper }
240     {
241         \msg_info:nnn { enumext } { package-load } { footnotehyper }
242     }
243     {
244         \typeout{No ~ footnotehyper ~ load}
245         \typeout{Load ~ and ~ use ~ \string\makesavenoteenv{enumext*}}
246         \bool_set_false:N \l__enumext_footnotes_key_bool
247     }
248 }

```

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.

```

249 \bool_if:NTF \l__enumext_hyperref_bool
250 {
251     \cs_new_eq:NN \__enumext_hypertarget:nn \hypertarget
252     \cs_new_eq:NN \__enumext_phantomsection: \phantomsection
253 }
254 {
255     \cs_new_eq:NN \__enumext_hypertarget:nn \use_none:nn
256     \cs_new_eq:NN \__enumext_phantomsection: \prg_do_nothing:
257 }
258 }

```

(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 `store-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.

```

259 \cs_new_protected:Npn \l__enumext_newlabel:nn #1 #2
260 {
261   \protected@write \@auxout { }
262   {
263     \token_to_str:N \newlabel {#1}
264     {
265       {#2}
266       \bool_if:NT \l__enumext_hyperref_bool
267       { { \thepage } {#2} {#1} }
268       { }
269     }
270   }
271   \__enumext_hypertarget:nn {#1} { }
272   \__enumext_phantomsection:
273 }

```

(End of definition for `\l__enumext_newlabel:nn`.)

10.8 Definition of counters

```

\__enumext_define_counters:Nn
\__enumext_define_counters:cn

```

To create the necessary “*counters*” we must first make sure that they are not already defined by the user or a package such as `enumitem`, otherwise a error will be returned and the package loading will be aborted. The arguments taken by the function are:

#1: A token list `\l__enumext_counter_X_tl` for “*store*” the counter’s name.

#2: The counter’s name.

```

274 \cs_new_protected:Npn \__enumext_define_counters:Nn #1 #2
275 {
276   \cs_if_exist:cTF { c@ #2 }
277   { \msg_fatal:nnn { enumext } { counters } { #2 } }
278   {
279     \tl_set:Nn #1 { #2 }
280     \newcounter { #2 }
281   }
282 }

```

(End of definition for `__enumext_define_counters:Nn`.)

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.

```

enumXi    283 \__enumext_define_counters:Nn \l__enumext_counter_i_tl { enumXi }
enumXii   284 \__enumext_define_counters:Nn \l__enumext_counter_ii_tl { enumXii }
enumXiii  285 \__enumext_define_counters:Nn \l__enumext_counter_iii_tl { enumXiii }
enumXiv   286 \__enumext_define_counters:Nn \l__enumext_counter_iv_tl { enumXiv }
enumXvii  287 \__enumext_define_counters:Nn \l__enumext_counter_v_tl { enumXv }
enumXviii 288 \__enumext_define_counters:Nn \l__enumext_counter_vi_tl { enumXvi }
           289 \__enumext_define_counters:Nn \l__enumext_counter_vii_tl { enumXvii }
           290 \__enumext_define_counters:Nn \l__enumext_counter_viii_tl { enumXviii }

```

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

10.9 Definition of labels

This part of the code is inspired by the `enumitem` package. The idea is to be able to access the counters using `\arabic*`, `\Alph*`, `\alph*`, `\Roman*` and `\roman*` to use them in the `label` key.

```
\__enumext_register_counter_style:Nn
```

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

```

291 \cs_new_protected:Npn \__enumext_register_counter_style:Nn #1 #2
292 {
293   \tl_const:cn { c__enumext_widest_ \cs_to_str:N #1 _tl } {#2}
294   \tl_gput_right:Nn \g__enumext_counter_styles_tl {#1}
295 }
296 \__enumext_register_counter_style:Nn \arabic { 0 }
297 \__enumext_register_counter_style:Nn \Alph { M }
298 \__enumext_register_counter_style:Nn \alph { m }
299 \__enumext_register_counter_style:Nn \Roman { VIII }
300 \__enumext_register_counter_style:Nn \roman { viii }

```

(End of definition for `__enumext_register_counter_style:Nn`.)

`__enumext_label_width_by_box:Nn`
`__enumext_label_width_by_box:cv`

The function `__enumext_label_width_by_box:Nn` set the default `\labelwidth` using a box width if no `labelwidth` key is passed.

```
301 \cs_new_protected:Npn \__enumext_label_width_by_box:Nn #1#2
302 {
303   \hbox_set:Nn \l__enumext_label_width_by_box {#2}
304   \dim_set:Nn #1 { \box_wd:N \l__enumext_label_width_by_box }
305 }
306 \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.

```
307 \cs_new_protected:Npn \__enumext_label_style:Nnn #1 #2 #3
308 {
309   \tl_clear_new:N #1
310   \tl_put_right:Ne #1 { \tl_trim_spaces:n {#3} }
311   \tl_gset_eq:NN \g__enumext_widest_label_tl #1
312   \tl_map_inline:Nn \g__enumext_counter_styles_tl
313   {
314     \tl_replace_all:Nne #1 { ##1* } { \exp_not:N ##1 {#2} }
315     \tl_greplace_all:Nne \g__enumext_widest_label_tl { ##1* }
316     { \tl_use:c { c__enumext_widest_ \cs_to_str:N ##1 _tl } }
317   }
318   \__enumext_label_width_by_box:Nn \l__enumext_current_widest_dim
319   { \tl_use:N \g__enumext_widest_label_tl }
320   \tl_set_eq:cN { the #2 } #1
321 }
322 \cs_generate_variant:Nn \__enumext_label_style:Nnn { cvn }
```

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

10.10 Setting keys associated with label

`font`
`labelsep`
`labelwidth`
`wrap-label`
`wrap-label*`

Definition of keys `font`, `labelsep`, `labelwidth`, `wrap-label` and `wrap-label*` keys for `enumext` and `keyans` environments.

```
323 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
324 {
325   \keys_define:nn { enumext / #1 }
326   {
327     font      .tl_set:c   = { l__enumext_label_font_style_#2_tl },
328     font      .value_required:n = true,
329     labelsep  .dim_set:c   = { l__enumext_labelsep_#2_dim },
330     labelsep  .initial:n   = {0.3333em},
331     labelsep  .value_required:n = true,
332     labelwidth .dim_set:c   = { l__enumext_labelwidth_#2_dim },
333     labelwidth .value_required:n = true,
334     wrap-label .cs_set_protected:cp = { __enumext_wrapper_label_#2:n } ##1,
335     wrap-label .initial:n   = {##1},
336     wrap-label .value_required:n = true,
337     wrap-label* .code:n = {
338       \bool_set_true:c { l__enumext_wrap_label_opt_#2_bool }
339       \keys_set:nn { enumext / #1 } { wrap-label = {##1} }
340     },
341     wrap-label* .value_required:n = true,
342   }
343 }
344 \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }
```

(End of definition for `font` and others.)

- 🔗 In this point, the following are set `__enumext_wrapper_label_X:n` which will be used by `__enumext_make_label:` for the different levels of the `enumext` environment and is set to `__enumext_wrapper_label_v:n` which will be used by `__enumext_keyans_make_label:` for `keyans` and `keyanspic` environments.

`align` The `align` key is implemented differently for “starred” and “non starred” environments.

```

345 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
346 {
347   \keys_define:nn { enumext / #1 }
348   {
349     align .choice:,
350     align / left .code:n =
351       {
352         \tl_clear:c { l__enumext_label_fill_left_#2_tl }
353         \tl_set:cn { l__enumext_label_fill_right_#2_tl } { \hfill }
354       },
355     align / right .code:n =
356       {
357         \tl_set:cn { l__enumext_label_fill_left_#2_tl } { \hfill }
358         \tl_clear:c { l__enumext_label_fill_right_#2_tl }
359       },
360     align / center .code:n =
361       {
362         \tl_set:cn { l__enumext_label_fill_left_#2_tl } { \hfill }
363         \tl_set:cn { l__enumext_label_fill_right_#2_tl } { \hfill }
364       },
365     align .initial:n = left,
366     align .value_required:n = true,
367   }
368 }
369 \clist_map_inline:nn
370 {
371   {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {keyans}{v}
372 }
373 { \__enumext_tmp:nn #1 }

```

Definition of `align` key for `enumext*` and `keyans*` environments.

```

374 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
375 {
376   \keys_define:nn { enumext / #1 }
377   {
378     align .choice:,
379     align / left .code:n = \str_set:cn { l__enumext_align_label_#2_str } { l },
380     align / right .code:n = \str_set:cn { l__enumext_align_label_#2_str } { r },
381     align / center .code:n = \str_set:cn { l__enumext_align_label_#2_str } { c },
382     align .initial:n = left,
383     align .value_required:n = true,
384   }
385 }
386 \clist_map_inline:nn { {enumext*}{vii}, {keyans*}{viii} } { \__enumext_tmp:nn #1 }

```

(End of definition for `align`.)

10.11 Setting label and ref keys

`__enumext_regex_label_ref_key:` The internal function `__enumext_regex_label_ref_key:` replace the `*` with the actual counter of the running level and is used by the `__enumext_set_label_ref:n` function.

It loops through the defined counter styles in `\c__enumext_counter_style_tl` and replace `*` by real command, for example, looking for `\arabic*` and replacing that by `\arabic{<counter>}` defined on the current level.

```

387 \cs_new_protected:Nn \__enumext_regex_label_ref_key:
388 {
389   \tl_map_inline:Nn \c__enumext_counter_style_tl
390   {
391     \regex_replace_once:nnN { \c{##1}\* }
392     { \c{##1}\cB{\u{l__enumext_ref_aux_tl}\cE} } \l__enumext_ref_key_arg_tl
393   }
394 }

```

(End of definition for `__enumext_regex_label_ref_key:`.)

`__enumext_set_label_ref:n` The `__enumext_set_label_ref:n` function controlled by the `ref` key is in charge of handling the customization of the reference system.

First we will set the variable `\l__enumext_the_counter_X_tl` according to the command created for *each counter*, apply the `regex` function `__enumext_regex_label_ref_key:` and then renew the command and save it in the variable `\l__enumext_counter_style_for_ref_X_tl`.

```

395 \cs_new_protected:Npn \__enumext_set_label_ref:n #1
396 {
397   \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
398   \tl_set_eq:Nc \l__enumext_ref_aux_tl { \l__enumext_counter_ \l__enumext_level: _tl }
399   \__enumext_regex_label_ref_key:
400   \tl_set_eq:Nc \l__enumext_ref_aux_tl { \l__enumext_the_counter_ \l__enumext_level: _tl }
401   \tl_put_right:ce { \l__enumext_counter_style_for_ref_ \l__enumext_level: _tl }
402   {
403     \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_aux_tl }
404     { \exp_not:V \l__enumext_ref_key_arg_tl }
405   }
406 }

```

(End of definition for __enumext_set_label_ref:n.)

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

```

407 \cs_new_protected:Npn \__enumext_use_key_ref:
408 {
409   \tl_if_empty:cF { \l__enumext_counter_style_for_ref_ \l__enumext_level: _tl }
410   {
411     \tl_use:c { \l__enumext_counter_style_for_ref_ \l__enumext_level: _tl }
412   }
413 }

```

(End of definition for __enumext_use_key_ref:.)

For `enumext*` and `keyans*` environments the situation is a bit different since `hyperref` interferes here (I am not clear why), so we will define a new function to execute the task.

To handle that we will look at the nesting level of the starred environments, later I will run the constraint functions to make everything OK.

__enumext_set_label_ref_h:n The __enumext_set_label_ref_h:n function controlled by the `ref` key is in charge of handling the customization of the reference system.

First we will set the variable \l__enumext_the_counter_X_tl according to the command created for *each counter*, apply the `regex` function __enumext_regex_label_ref_key: and then renew the command and save it in the variable \l__enumext_counter_style_for_ref_X_tl.

```

414 \cs_new_protected:Npn \__enumext_set_label_ref_h:n #1
415 {
416   \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
417   \int_compare:nNnTF { \l__enumext_level_h_int } = { 1 }
418   {
419     \tl_set_eq:NN \l__enumext_ref_aux_tl \l__enumext_counter_vii_tl
420     \__enumext_regex_label_ref_key:
421     \tl_set_eq:NN \l__enumext_ref_aux_tl \l__enumext_the_counter_vii_tl
422     \tl_put_right:Ne \l__enumext_counter_style_for_ref_vii_tl
423     {
424       \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_aux_tl }
425       { \exp_not:V \l__enumext_ref_key_arg_tl }
426     }
427   }
428   {
429     \tl_set_eq:NN \l__enumext_ref_aux_tl \l__enumext_counter_viii_tl
430     \__enumext_regex_label_ref_key:
431     \tl_set_eq:NN \l__enumext_ref_aux_tl \l__enumext_the_counter_viii_tl
432     \tl_put_right:Ne \l__enumext_counter_style_for_ref_vii_tl
433     {
434       \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_aux_tl }
435       { \exp_not:V \l__enumext_ref_key_arg_tl }
436     }
437   }
438 }

```

(End of definition for __enumext_set_label_ref_h:n.)

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

```

439 \cs_new_protected:Npn \__enumext_use_key_ref_h:
440 {
441   \int_compare:nNnTF { \l__enumext_level_h_int } = { 1 }
442   {

```

```

443         \tl_if_empty:NF \l__enumext_counter_style_for_ref_vii_tl
444         {
445             \tl_use:N \l__enumext_counter_style_for_ref_vii_tl
446         }
447     }
448 {
449     \tl_if_empty:NF \l__enumext_counter_style_for_ref_viii_tl
450     {
451         \tl_use:N \l__enumext_counter_style_for_ref_viii_tl
452     }
453 }
454 }

```

(End of definition for `__enumext_use_key_ref_h:`.)

10.11.1 Define and set label key for enumext environment

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

```

\__enumext_label_i_tl
\__enumext_label_ii_tl
\__enumext_label_iii_tl
\__enumext_label_iv_tl
455 \cs_set_protected:Npn \__enumext_tmp:nnn #1 #2 #3
456 {
457     \keys_define:nn { enumext / #1 }
458     {
459         label .code:n = {
460             \__enumext_label_style:cvn { \__enumext_label_#2_tl }
461             { \__enumext_counter_#2_tl } {##1}
462             \dim_set_eq:cN { \__enumext_labelwidth_#2_dim }
463             \l__enumext_current_widest_dim
464         },
465         label .initial:n = #3,
466         label .value_required:n = true,
467         ref .code:n = \__enumext_set_label_ref:n {##1},
468         ref .value_required:n = true,
469     }
470 }
471 \__enumext_tmp:nnn { level-1 } { i } { \arabic*. }
472 \__enumext_tmp:nnn { level-2 } { ii } { (\alph*. ) }
473 \__enumext_tmp:nnn { level-3 } { iii } { \roman*. }
474 \__enumext_tmp:nnn { level-4 } { iv } { \Alph*. }

```

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

10.11.2 Define and set label key for enumext* and keyans* environments

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

```

\__enumext_label_vii_tl
\__enumext_label_viii_tl
475 \cs_set_protected:Npn \__enumext_tmp:nnn #1 #2 #3
476 {
477     \keys_define:nn { enumext / #1 }
478     {
479         label .code:n = {
480             \__enumext_label_style:cvn { \__enumext_label_#2_tl }
481             { \__enumext_counter_#2_tl } {##1}
482             \dim_set_eq:cN { \__enumext_labelwidth_#2_dim }
483             \l__enumext_current_widest_dim
484         },
485         label .initial:n = #3,
486         label .value_required:n = true,
487         ref .code:n = \__enumext_set_label_ref_h:n {##1},
488         ref .value_required:n = true,
489     }
490 }
491 \__enumext_tmp:nnn { enumext* } { vii } { \arabic*. }
492 \__enumext_tmp:nnn { keyans* } { viii } { (\Alph*. ) }

```

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

10.11.3 Define and set label key for keyans and keyanspic environment

Here we set the default *label* for `keyans` and `keyanspic` environment, along with the default value for `labelwidth`. The `keyanspic` environment use the same *label* as the `keyans` environment.

Define and set `label` key for `keyans` environment.

```

493 \keys_define:nn { enumext / keyans }
494 {

```



```

495     label .code:n = {
496         \__enumext_label_style:cvn { \__enumext_label_v_tl }
497         { \__enumext_counter_v_tl } {#1}
498         \dim_set_eq:cN { \__enumext_labelwidth_v_dim }
499         \__enumext_current_widest_dim
500         \__enumext_label_style:cvn { \__enumext_label_vi_tl }
501         { \__enumext_counter_vi_tl } {#1}
502         \dim_set_eq:cN { \__enumext_labelwidth_v_dim }
503         \__enumext_current_widest_dim
504     },
505     label .initial:n = (\Alph*),
506     label .value_required:n = true,
507 }

```

(End of definition for `label`, `__enumext_label_v_tl`, and `__enumext_label_vi_tl`.)

10.12 Setting start and widest keys

```

\__enumext_start_from:NNn
\__enumext_start_from:ccn

```

The function `__enumext_start_from:NNn` used by the `start` key take three arguments:

```

#1: \__enumext_label_X_tl
#2: \__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.

```

508 \cs_new_protected:Npn \__enumext_start_from:NNn #1 #2 #3
509 {
510     \__enumext_if_is_int:nTF { #3 }
511     {
512         \int_set:Nn #2 {#3}
513     }
514     {
515         \regex_match:nVT { \c{Alph} | \c{alph} } {#1}
516         { \int_set:Nn #2 { \int_from_alph:n {#3} } }
517         \regex_match:nVT { \c{Roman} | \c{roman} } {#1}
518         { \int_set:Nn #2 { \int_from_roman:n {#3} } }
519     }
520 }
521 \cs_generate_variant:Nn \__enumext_start_from:NNn { ccn }

```

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

```

\__enumext_widest_from:nNNn
\__enumext_widest_from:nccn

```

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

```

#1: The counter associated with the environment level
#2: \__enumext_label_X_tl
#3: \__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.

```

522 \cs_new_protected:Npn \__enumext_widest_from:nNNn #1 #2 #3 #4
523 {
524     \__enumext_if_is_int:nTF {#4}
525     {
526         \setcounter{enumX#1} { #4 }
527     }
528     {
529         \regex_match:nVT { \c{Alph} | \c{alph} } {#2}
530         { \setcounter{enumX#1} { \int_from_alph:n {#4} } }
531         \regex_match:nVT { \c{Roman} | \c{roman} } {#2}
532         { \setcounter{enumX#1} { \int_from_roman:n {#4} } }
533     }
534     \__enumext_label_width_by_box:cv
535     { \__enumext_labelwidth_#1_dim } { \__enumext_label_#1_tl }
536 }
537 \cs_generate_variant:Nn \__enumext_widest_from:nNNn { nccn }

```

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

```

start
widest
\l__enumext_start_X_int
538 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
539 {
540   \keys_define:nn { enumext / #1 }
541   {
542     start .code:n = {
543       \__enumext_start_from:ccn
544       { l__enumext_label_#2_tl }
545       { l__enumext_start_#2_int } {##1}
546     },
547     start .initial:n = 1,
548     widest .code:n = {
549       \__enumext_widest_from:nccn {#2}
550       { l__enumext_label_#2_tl }
551       { l__enumext_labelwidth_#2_dim } {##1}
552     },
553     widest .value_required:n = true,
554     start .value_required:n = true,
555   }
556 }
557 \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }

```

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

10.13 Setting keys for vertical spaces

Define and set topsep, partopsep, parsep, itemsep, noitemsep and nosep keys for `enumext` and `keyans` environments.

```

topsep
partopsep
parsep
noitemsep
nosep
558 \cs_set_protected:Npn \__enumext_tmp:nnnnnn #1 #2 #3 #4 #5 #6
559 {
560   \keys_define:nn { enumext / #1 }
561   {
562     topsep .skip_set:c = { l__enumext_topsep_#2_skip },
563     topsep .initial:n = {#3},
564     topsep .value_required:n = true,
565     partopsep .skip_set:c = { l__enumext_partopsep_#2_skip },
566     partopsep .initial:n = {#4},
567     partopsep .value_required:n = true,
568     parsep .skip_set:c = { l__enumext_parsep_#2_skip },
569     parsep .initial:n = {#5},
570     parsep .value_required:n = true,
571     itemsep .skip_set:c = { l__enumext_itemsep_#2_skip },
572     itemsep .initial:n = {#6},
573     itemsep .value_required:n = true,
574     noitemsep .meta:n = { itemsep = 0pt, parsep = 0pt },
575     noitemsep .value_forbidden:n = true,
576     nosep .meta:n = {
577       itemsep = 0pt, parsep = 0pt,
578       topsep = 0pt, partopsep = 0pt,
579     },
580     nosep .value_forbidden:n = true,
581   }
582 }

```

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

```

583 \__enumext_tmp:nnnnnn { level-1 } { i } { 8.0pt plus 2.0pt minus 4.0pt }
584 { 2.0pt plus 1.0pt minus 1.0pt } { 4.0pt plus 2.0pt minus 1.0pt }
585 { 4.0pt plus 2.0pt minus 1.0pt }
586 \__enumext_tmp:nnnnnn { level-2 } { ii } { 4.0pt plus 2.0pt minus 1.0pt }
587 { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
588 { 2.0pt plus 1.0pt minus 1.0pt }
589 \__enumext_tmp:nnnnnn { level-3 } { iii } { 2.0pt plus 1.0pt minus 1.0pt }
590 { 1.0pt minus 1.0pt } { 0pt } { 2.0pt plus 1.0pt minus 1.0pt }
591 \__enumext_tmp:nnnnnn { level-4 } { iv } { 2.0pt plus 1.0pt minus 1.0pt }
592 { 1.0pt minus 1.0pt } { 0pt } { 2.0pt plus 1.0pt minus 1.0pt }
593 \__enumext_tmp:nnnnnn { keyans } { v } { 4.0pt plus 2.0pt minus 1.0pt }
594 { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
595 { 2.0pt plus 1.0pt minus 1.0pt }
596 \__enumext_tmp:nnnnnn { enumext* } { vii } { 8.0pt plus 2.0pt minus 4.0pt }
597 { 2.0pt plus 1.0pt minus 1.0pt } { 4.0pt plus 2.0pt minus 1.0pt }
598 { 4.0pt plus 2.0pt minus 1.0pt }

```

```

599 \__enumext_tmp:nnnnnn { keyans* } { viii } { 4.0pt plus 2.0pt minus 1.0pt }
600 { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
601 { 2.0pt plus 1.0pt minus 1.0pt }

```

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

10.14 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` and `keyans` environments.

```

602 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
603 {
604   \keys_define:nn { enumext / #1 }
605   {
606     itemindent .dim_set:c = { l__enumext_fake_item_indent_#2_dim },
607     itemindent .value_required:n = true,
608     rightmargin .dim_set:c = { l__enumext_rightmargin_#2_dim },
609     rightmargin .value_required:n = true,
610     listparindent .dim_set:c = { l__enumext_listparindent_#2_dim },
611     listparindent .value_required:n = true,
612     list-offset .dim_set:c = { l__enumext_listoffset_#2_dim },
613     list-offset .value_required:n = true,
614     list-indent .code:n =
615       \bool_set_true:c { l__enumext_leftmargin_tmp_#2_bool }
616       \dim_set:cn { l__enumext_leftmargin_tmp_#2_dim } {##1},
617     list-indent .value_required:n = true,
618   }
619 }
620 \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }

```

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

For `enumext*` and `keyans*` environments the situation is a bit different, the `list-indent` key behaves like the `list-offset` key.

```

621 \cs_set_protected:Npn \__enumext_tmp:n #1
622 {
623   \keys_define:nn { enumext / #1 } { list-indent .initial:n = 0pt, }
624 }
625 \clist_map_inline:nn { enumext*, keyans* } { \__enumext_tmp:n {#1} }

```

10.14.1 Functions for setting the fake itemindent

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.

```

626 \cs_set_protected:Nn \__enumext_fake_item:
627 {
628   \dim_compare:nNnT
629     { \dim_use:c { l__enumext_fake_item_indent_ \__enumext_level: _dim } }
630     >
631     { \c_zero_dim }
632   {
633     \tl_set:ce { l__enumext_fake_item_indent_ \__enumext_level: _tl }
634     {
635       \exp_not:N \mode_leave_vertical:
636       \exp_not:n { \skip_horizontal:n }
637       { \dim_use:c { l__enumext_fake_item_indent_ \__enumext_level: _dim } }
638       \ignorespaces
639     }
640   }
641 }
642 \cs_set_protected:Nn \__enumext_keyans_fake_item:
643 {
644   \dim_compare:nNnT
645     { \l__enumext_fake_item_indent_v_dim } > { \c_zero_dim }
646     {
647       \tl_set:Ne \l__enumext_fake_item_indent_v_tl
648       {
649         \exp_not:N \mode_leave_vertical:
650         \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_v_dim
651       }
652     }
653 }

```

```

653     }
654     \cs_set_protected:Nn \__enumext_fake_item_vii:
655     {
656         \dim_compare:nNnT
657         { \l__enumext_fake_item_indent_vii_dim } > { \c_zero_dim }
658         {
659             \tl_set:Nc \l__enumext_fake_item_indent_vii_tl
660             {
661                 \exp_not:N \mode_leave_vertical:
662                 \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_vii_dim
663             }
664         }
665     }
666     \cs_set_protected:Nn \__enumext_fake_item_viii:
667     {
668         \dim_compare:nNnT
669         { \l__enumext_fake_item_indent_viii_dim } > { \c_zero_dim }
670         {
671             \tl_set:Nc \l__enumext_fake_item_indent_viii_tl
672             {
673                 \exp_not:N \mode_leave_vertical:
674                 \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_viii_dim
675             }
676         }
677     }

```

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

10.15 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.

```

678 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
679 {
680     \keys_define:nn { enumext / #1 }
681     {
682         show-length .bool_set:c = { \l__enumext_show_length_#2_bool },
683         show-length .initial:n = false,
684     }
685 }
686 \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }

```

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

10.16 Setting before, after and first keys

before

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

before*

after

first

```

687 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
688 {
689     \keys_define:nn { enumext / #1 }
690     {
691         before .tl_set:c = { \l__enumext_before_no_starred_key_#2_tl },
692         before .value_required:n = true,
693         before* .tl_set:c = { \l__enumext_before_starred_key_#2_tl },
694         before* .value_required:n = true,
695         after .tl_set:c = { \l__enumext_after_stop_list_#2_tl },
696         after .value_required:n = true,
697         first .tl_set:c = { \l__enumext_after_list_args_#2_tl },
698         first .value_required:n = true,
699     }
700 }
701 \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }

```

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

10.16.1 Functions for before, after and first keys in enumext

The function `__enumext_before_args_exec:` executes the `{\code}` set by the `before*` key “before” the `enumext` environment is started. The `{\code}` is executed “without” knowing any definition of the *second argument* of the list.

```

702 \cs_new_protected:Nn \__enumext_before_args_exec:

```

```

703 {
704   \tl_use:c { l__enumext_before_starred_key_ \__enumext_level: _tl }
705 }

```

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⟩`.

```

706 \cs_new_protected:Nn \__enumext_before_keys_exec:
707 {
708   \tl_use:c { l__enumext_before_no_starred_key_ \__enumext_level: _tl }
709 }

```

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

```

710 \cs_new_protected:Nn \__enumext_after_stop_list:
711 {
712   \tl_use:c { l__enumext_after_stop_list_ \__enumext_level: _tl }
713 }

```

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`.

```

714 \cs_new_protected:Nn \__enumext_after_args_exec:
715 {
716   \tl_use:c { l__enumext_after_list_args_ \__enumext_level: _tl }
717 }

```

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

10.16.2 Functions for before, after and first keys in keyans

```

\__enumext_before_args_exec_v:
\__enumext_before_keys_exec_v:
\__enumext_after_stop_list_v:
\__enumext_after_args_exec_v:

```

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

```

718 \cs_new_protected:Nn \__enumext_before_args_exec_v:
719 {
720   \tl_use:N \l__enumext_before_starred_key_v_tl
721 }

```

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

```

722 \cs_new_protected:Nn \__enumext_before_keys_exec_v:
723 {
724   \tl_use:N \l__enumext_before_no_starred_key_v_tl
725 }

```

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

```

726 \cs_new_protected:Nn \__enumext_after_stop_list_v:
727 {
728   \tl_use:N \l__enumext_after_stop_list_v_tl
729 }

```

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

```

730 \cs_new_protected:Nn \__enumext_after_args_exec_v:
731 {
732   \tl_use:N \l__enumext_after_list_args_v_tl
733 }

```

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

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

```

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

```

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

```

734 \cs_new_protected:Nn \__enumext_before_args_exec_vii:
735 {
736   \tl_use:N \l__enumext_before_starred_key_vii_tl
737 }
738 \cs_new_protected:Nn \__enumext_before_args_exec_viii:
739 {
740   \tl_use:N \l__enumext_before_starred_key_viii_tl
741 }

```

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

```

742 \cs_new_protected:Nn \__enumext_before_keys_exec_vii:
743 {
744   \tl_use:N \l__enumext_before_no_starred_key_vii_tl
745 }
746 \cs_new_protected:Nn \__enumext_before_keys_exec_viii:
747 {
748   \tl_use:N \l__enumext_before_no_starred_key_viii_tl
749 }

```

The function `__enumext_after_stop_list:` executes the `{\code}` set by the `after` key “after” the `keyans` environment has finished.

```

750 \cs_new_protected:Nn \__enumext_after_stop_list_vii:
751 {
752   \tl_use:N \l__enumext_after_stop_list_vii_tl
753 }
754 \cs_new_protected:Nn \__enumext_after_stop_list_viii:
755 {
756   \tl_use:N \l__enumext_after_stop_list_viii_tl
757 }

```

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

```

758 \cs_new_protected:Nn \__enumext_after_args_exec_vii:
759 {
760   \tl_use:N \l__enumext_after_list_args_vii_tl
761 }
762 \cs_new_protected:Nn \__enumext_after_args_exec_viii:
763 {
764   \tl_use:N \l__enumext_after_list_args_viii_tl
765 }

```

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

10.17 Setting keys for multicol and minipage

`mini-env` 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.
`mini-sep` Define and set `mini-env`, `mini-sep`, `columns-sep` and `columns` keys for `enumext` and `keyans` environments.
`columns-sep`
`columns`

```

766 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
767 {
768   \keys_define:nn { enumext / #1 }
769   {
770     mini-env .dim_set:c = { \l__enumext_minipage_right_#2_dim },
771     mini-env .value_required:n = true,
772     mini-sep .dim_set:c = { \l__enumext_minipage_hsep_#2_dim },
773     mini-sep .initial:n = 0.3333em,
774     mini-sep .value_required:n = true,
775     columns-sep .dim_set:c = { \l__enumext_columns_sep_#2_dim },
776     columns-sep .value_required:n = true,
777     columns .int_set:c = { \l__enumext_columns_#2_int },
778     columns .initial:n = 1,
779     columns .value_required:n = true,
780   }
781 }
782 \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 default value for `columns` key are 2 and the command `\miniright` is not available, so we will add the keys `miniright` and `miniright*` to implement support for `minipage`.

```

783 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
784 {
785   \keys_define:nn { enumext / #1 }
786   {
787     columns .initial:n = 2,
788     miniright .tl_gset:c = { g__enumext_miniright_code_#2_tl },
789     miniright .value_required:n = true,
790     miniright* .code:n = {

```

```

791                                     \bool_gset_true:c { g__enumext_minipage_center_#2_bool }
792                                     \keys_set:nn { enumext / #1 } { miniright = {##1} }
793                                     },
794                                     miniright* .value_required:n = true,
795                                     }
796                                     }
797 \clist_map_inline:nn { {enumext*}{vii}, {keyans*}{viii} } { \__enumext_tmp:nn #1 }

```

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

10.18 Adjustment of vertical spaces for multicol

When nesting a “*list environment*” inside the `multicol` environment, the values of the “*vertical spaces*” are lost, basically the `multicol` environment takes control over them. Graphically it can be seen like in the figure 7.

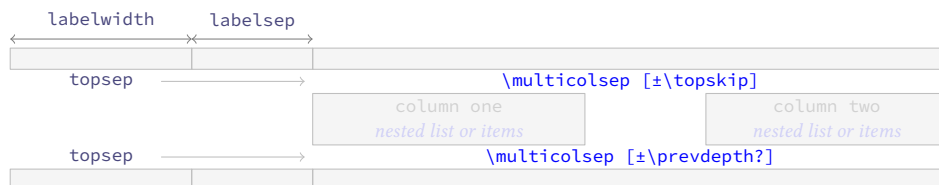


Figure 7: Representation of the vertical space in `multicol` for a nested level.

To keep the desired spaces *above* and *below* in the “*list environment*” (`\topsep` + `[\partopsep]`) it is necessary to “*adjust*” the spaces added by the `multicol` 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.

10.18.1 Adjustment of vertical spaces for multicol 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 `multicol` 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_multicol_above_X_skip` and `\l__enumext_multicol_below_X_skip` equal to the value of `\topsep` in the *current level*.

```

798 \cs_new_protected:Nn \__enumext_multi_set_vskip:
799 {
800   \skip_set:cn { \l__enumext_multicol_above_ \__enumext_level: _skip }
801   {
802     \skip_use:c { \l__enumext_topsep_ \__enumext_level: _skip }
803   }
804   \skip_set:cn { \l__enumext_multicol_below_ \__enumext_level: _skip }
805   {
806     \skip_use:c { \l__enumext_topsep_ \__enumext_level: _skip }
807   }
808   \__enumext_add_pre_parsep:
809 }

```

(End of definition for `__enumext_multi_set_vskip:`)

`__enumext_add_pre_parsep:` The function `__enumext_add_pre_parsep:` “*adjusted*” the value of `\l__enumext_multicol_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*.

```

810 \cs_new_protected:Nn \__enumext_add_pre_parsep:
811 {
812   \int_case:nn { \l__enumext_level_int }
813   {
814     { 2 }{
815       \skip_if_eq:nnF { \l__enumext_parsep_i_skip } { \c_zero_skip }
816       {
817         \skip_add:Nn \l__enumext_multicol_above_ii_skip { \l__enumext_parsep_i_skip }
818       }
819     }
820     { 3 }{
821       \skip_if_eq:nnF { \l__enumext_parsep_ii_skip } { \c_zero_skip }
822       {

```



```

823         \skip_add:Nn \l__enumext_multicols_above_iii_skip { \l__enumext_parsep_iii_skip
824     }
825 }
826 { 4 }{
827     \skip_if_eq:nnF { \l__enumext_parsep_iii_skip } { \c_zero_skip }
828     {
829         \skip_add:Nn \l__enumext_multicols_above_iv_skip { \l__enumext_parsep_iii_skip
830     }
831 }
832 }
833 }

```

(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*.

```

834 \cs_new_protected:Nn \__enumext_multi_addvspace:
835 {
836     \__enumext_multi_set_vskip:
837     \mode_if_vertical:T
838     {
839         \skip_add:cn { \l__enumext_multicols_above_ \__enumext_level: _skip }
840         {
841             \skip_use:c { \l__enumext_partopsep_ \__enumext_level: _skip }
842         }
843         \skip_add:cn { \l__enumext_multicols_below_ \__enumext_level: _skip }
844         {
845             \skip_use:c { \l__enumext_partopsep_ \__enumext_level: _skip }
846         }
847     }
848     \par\nopagebreak
849     \addvspace{ \skip_use:c { \l__enumext_multicols_above_ \__enumext_level: _skip } }
850 }

```

(End of definition for `__enumext_multi_addvspace:`)

10.18.2 Adjustment of vertical spaces for multicols in keyans

`__enumext_keyans_multi_set_vskip:` 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`.

`__enumext_keyans_multi_addvspace:`

```

851 \cs_new_protected:Nn \__enumext_keyans_multi_set_vskip:
852 {
853     \skip_set:Nn \l__enumext_multicols_above_v_skip
854     {
855         \l__enumext_topsep_v_skip
856     }
857     \skip_set:Nn \l__enumext_multicols_below_v_skip
858     {
859         \l__enumext_topsep_v_skip
860     }
861 }
862 \cs_new_protected:Nn \__enumext_keyans_multi_addvspace:
863 {
864     \__enumext_keyans_multi_set_vskip:
865     \mode_if_vertical:T
866     {
867         \skip_add:Nn \l__enumext_multicols_above_v_skip
868         {
869             \skip_use:N \l__enumext_partopsep_v_skip
870         }
871         \skip_add:Nn \l__enumext_multicols_below_v_skip
872         {
873             \skip_use:N \l__enumext_partopsep_v_skip
874         }
875     }
876     \par\nopagebreak
877     \addvspace{ \l__enumext_multicols_above_v_skip }
878 }

```

(End of definition for `__enumext_keyans_multi_set_vskip:` and `__enumext_keyans_multi_addvspace:`)

10.19 Adjustment of vertical spaces for minipage

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



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

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

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

Again I must make clear that the implementation here is a “bit questionable”, but hunting the spaces (`glue`) produced by the `minipage` environment is quite complicated, even more if `multicols` it is nested. The setting of the values was more “trial and error” (aprox to `\strutbox`), using the help of the `lua-visual-debug`[12] package, again my attempts to find the correct values using `\showoutput` and `\showboxdepth` absolutely failed.

`__enumext_mini_env*` Creates a `__enumext_mini_env*` environment (custom version of `minipage`) setting the `\if@minipage` switch to “false” to allow spaces at the “above” of the environment, plus we will add `\vspace{0pt}` to maintain alignment on “top”. This environment will be used internally by the `mini-env` key, it is not documented in the user interface and is for internal use only.

```

879 \DeclareDocumentEnvironment{__enumext_mini_env*}{ m }
880 {
881   \__enumext_minipage:w [ t ] { #1 }
882   \legacy_if_gset_false:n { @minipage }
883   \vspace { 0pt }
884 }
885 { \__enumext_endminipage: }
```

(End of definition for `__enumext_mini_env*`.)

10.19.1 Adjustment of vertical spaces for minipage in enumext

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

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

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

```

886 \cs_new_protected:Nn \__enumext_mini_set_vskip:
887 {
888   \int_compare:nNnTF
889     { \int_use:c { \l__enumext_columns_ \__enumext_level: _int } } > { 1 }
890     {
```

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

```

891   \skip_if_eq:nnTF
892     { \skip_use:c { \l__enumext_topsep_ \__enumext_level: _skip } } { \c_zero_skip }
893     {
894       \skip_set:Nn \l__enumext_minipage_left_skip
895         {
896           -0.150\box_dp:N \strutbox
897         }
898       \skip_set:Nn \l__enumext_minipage_right_skip
899         {
900           0.695\box_dp:N \strutbox
901         }

```

```

902     \skip_set:Nn \l__enumext_minipage_after_skip
903     {
904         \box_dp:N \strutbox
905     }
906     \__enumext_zero_parsep:
907 }
908 {
909     \skip_set:Nn \l__enumext_minipage_left_skip
910     {
911         \skip_use:c { \l__enumext_topsep_ \__enumext_level: _skip }
912     }
913     \skip_set:Nn \l__enumext_minipage_right_skip
914     {
915         0.695\box_dp:N \strutbox
916     }
917     \skip_set:Nn \l__enumext_minipage_after_skip
918     {
919         1.85\box_dp:N \strutbox
920         + \skip_use:c { \l__enumext_topsep_ \__enumext_level: _skip }
921     }
922 }
923 }
924 {

```

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

```

925     \skip_if_eq:nnTF
926     { \skip_use:c { \l__enumext_topsep_ \__enumext_level: _skip } } { \c_zero_skip }
927     {
928         \skip_set:Nn \l__enumext_minipage_left_skip
929         {
930             0.5\box_dp:N \strutbox
931             - \skip_use:c { \l__enumext_partopsep_ \__enumext_level: _skip }
932         }
933         \skip_set:Nn \l__enumext_minipage_right_skip
934         {
935             \skip_use:c { \l__enumext_partopsep_ \__enumext_level: _skip }
936         }
937         \skip_set:Nn \l__enumext_minipage_after_skip
938         {
939             1.6\box_dp:N \strutbox
940         }
941     }
942     {
943         \skip_set:Nn \l__enumext_minipage_left_skip
944         {
945             0.5875\box_dp:N \strutbox
946             - \skip_use:c { \l__enumext_partopsep_ \__enumext_level: _skip }
947         }
948         \skip_set:Nn \l__enumext_minipage_right_skip
949         {
950             + \skip_use:c { \l__enumext_topsep_ \__enumext_level: _skip }
951             + \skip_use:c { \l__enumext_partopsep_ \__enumext_level: _skip }
952         }
953         \skip_set:Nn \l__enumext_minipage_after_skip
954         {
955             0.325\box_dp:N \strutbox
956             + \skip_use:c { \l__enumext_topsep_ \__enumext_level: _skip }
957         }
958     }
959 }
960 }

```

(End of definition for `__enumext_mini_set_vskip:`)

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

```

961 \cs_new_protected:Nn \__enumext_zero_parsep:
962 {

```

```

963 \int_case:nn { \l__enumext_level_int }
964 {
965   { 2 }{
966     \skip_if_eq:nnF { \l__enumext_parsep_i_skip } { \c_zero_skip }
967     {
968       \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
969     }
970   }
971   { 3 }{
972     \skip_if_eq:nnF { \l__enumext_parsep_ii_skip } { \c_zero_skip }
973     {
974       \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
975     }
976   }
977   { 4 }{
978     \skip_if_eq:nnF { \l__enumext_parsep_iii_skip } { \c_zero_skip }
979     {
980       \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
981     }
982   }
983 }
984 }

```

(End of definition for `__enumext_zero_parsep:`.)

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

```

985 \cs_new_protected:Nn \__enumext_mini_addvspace:
986 {
987   \__enumext_mini_set_vskip:
988   \mode_if_vertical:T
989   {
990     \skip_add:Nn \l__enumext_minipage_left_skip
991     {
992       \skip_use:c { \l__enumext_partopsep_ \l__enumext_level: _skip }
993     }
994     \skip_add:Nn \l__enumext_minipage_after_skip
995     {
996       \skip_use:c { \l__enumext_partopsep_ \l__enumext_level: _skip }
997     }
998   }
999   \par\nopagebreak
1000   \addvspace { \l__enumext_minipage_left_skip }
1001 }

```

(End of definition for `__enumext_mini_addvspace:`.)

10.19.2 Adjustment of vertical spaces for minipage in keyans

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

```

1002 \cs_new_protected:Nn \__enumext_keyans_mini_set_vskip:
1003 {
1004   \skip_zero_new:N \l__enumext_minipage_after_skip
1005   \skip_zero_new:N \l__enumext_minipage_left_skip
1006   \skip_zero_new:N \l__enumext_minipage_right_skip
1007   \int_compare:nNnTF { \l__enumext_columns_v_int } > { 1 }
1008   {
1009     \skip_if_eq:nnTF { \l__enumext_topsep_v_skip } { \c_zero_skip }
1010     {
1011       \skip_set:Nn \l__enumext_minipage_left_skip { -0.25\box_dp:N \strutbox }
1012       \skip_set:Nn \l__enumext_minipage_right_skip { 0.705\box_dp:N \strutbox }
1013       \skip_set:Nn \l__enumext_minipage_after_skip { \box_dp:N \strutbox }
1014       \skip_if_eq:nnF { \l__enumext_parsep_i_skip } { \c_zero_skip }
1015       {
1016         \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
1017       }
1018     }
1019   }

```

```

1019     {
1020         \skip_set:Nn \l__enumext_minipage_left_skip
1021         {
1022             \skip_use:N \l__enumext_topsep_v_skip
1023         }
1024         \skip_set:Nn \l__enumext_minipage_right_skip
1025         {
1026             0.705\box_dp:N \strutbox
1027         }
1028         \skip_set:Nn \l__enumext_minipage_after_skip
1029         {
1030             1.85\box_dp:N \strutbox + \l__enumext_topsep_v_skip
1031         }
1032     }
1033 }
1034 {
1035     \skip_if_eq:nnTF { \l__enumext_topsep_v_skip } { \c_zero_skip }
1036     {
1037         \skip_set:Nn \l__enumext_minipage_left_skip
1038         {
1039             0.5\box_dp:N \strutbox
1040             + \l__enumext_partopsep_v_skip
1041         }
1042         \skip_set:Nn \l__enumext_minipage_right_skip
1043         {
1044             \l__enumext_partopsep_v_skip
1045         }
1046         \skip_set:Nn \l__enumext_minipage_after_skip { 1.6\box_dp:N \strutbox }
1047     }
1048     {
1049         \skip_set:Nn \l__enumext_minipage_left_skip
1050         {
1051             0.5875\box_dp:N \strutbox - \l__enumext_partopsep_v_skip
1052         }
1053         \skip_set:Nn \l__enumext_minipage_right_skip
1054         {
1055             \l__enumext_topsep_v_skip + \l__enumext_partopsep_v_skip
1056         }
1057         \skip_set:Nn \l__enumext_minipage_after_skip
1058         {
1059             0.325\box_dp:N \strutbox + \l__enumext_topsep_v_skip
1060         }
1061     }
1062 }
1063 }

```

(End of definition for `__enumext_keyans_mini_set_vskip:`)

`__enumext_keyans_mini_addvspace:`

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

```

1064 \cs_new_protected:Nn \__enumext_keyans_mini_addvspace:
1065 {
1066     \__enumext_keyans_mini_set_vskip:
1067     \mode_if_vertical:T
1068     {
1069         \skip_add:Nn \l__enumext_minipage_left_skip
1070         {
1071             \l__enumext_partopsep_v_skip
1072         }
1073         \skip_add:Nn \l__enumext_minipage_after_skip
1074         {
1075             \l__enumext_partopsep_v_skip
1076         }
1077     }
1078     \par\nopagebreak
1079     \addvspace { \l__enumext_minipage_left_skip }
1080 }

```

(End of definition for `__enumext_keyans_mini_addvspace:`)

10.19.3 Adjustment of vertical spaces for minipage in enumext* and keyans*

`__enumext_mini_set_vskip_vii:`
`__enumext_mini_set_vskip_viii:`

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

```

1081 \cs_new_protected:Nn \__enumext_mini_set_vskip_vii:
1082 {
1083   \skip_zero_new:N \l__enumext_minipage_left_skip
1084   \skip_gzero_new:N \g__enumext_minipage_right_skip
1085   \skip_gzero_new:N \g__enumext_minipage_after_skip
1086   \skip_if_eq:nnTF { \l__enumext_topsep_vii_skip } { \c_zero_skip }
1087   {
1088     \skip_set:Nn \l__enumext_minipage_left_skip { 0.5\box_dp:N \strutbox }
1089     \skip_gset:Nn \g__enumext_minipage_right_skip { 0.325\box_dp:N \strutbox }
1090   }
1091   {
1092     \skip_set:Nn \l__enumext_minipage_left_skip { 0.5875\box_dp:N \strutbox }
1093     \skip_gset:Nn \g__enumext_minipage_right_skip
1094     {
1095       \l__enumext_topsep_vii_skip
1096     }
1097     \skip_gset:Nn \g__enumext_minipage_after_skip
1098     {
1099       0.325\box_dp:N \strutbox + \l__enumext_topsep_vii_skip
1100     }
1101   }
1102 }
1103 \cs_new_protected:Nn \__enumext_mini_set_vskip_viii:
1104 {
1105   \skip_zero_new:N \l__enumext_minipage_after_skip
1106   \skip_zero_new:N \l__enumext_minipage_left_skip
1107   \skip_zero_new:N \l__enumext_minipage_right_skip
1108   \skip_if_eq:nnTF { \l__enumext_topsep_viii_skip } { \c_zero_skip }
1109   {
1110     \skip_set:Nn \l__enumext_minipage_left_skip
1111     {
1112       0.5\box_dp:N \strutbox
1113     }
1114     \skip_set:Nn \l__enumext_minipage_right_skip
1115     {
1116       \l__enumext_partopsep_viii_skip
1117     }
1118     \skip_set:Nn \l__enumext_minipage_after_skip
1119     {
1120       1.6\box_dp:N \strutbox
1121     }
1122   }
1123   {
1124     \skip_set:Nn \l__enumext_minipage_left_skip
1125     {
1126       0.5875\box_dp:N \strutbox
1127     }
1128     \skip_set:Nn \l__enumext_minipage_right_skip
1129     {
1130       \l__enumext_topsep_viii_skip
1131     }
1132     \skip_set:Nn \l__enumext_minipage_after_skip
1133     {
1134       0.325\box_dp:N \strutbox + \l__enumext_topsep_viii_skip
1135     }
1136   }
1137 }

```

(End of definition for `__enumext_mini_set_vskip_vii:` and `__enumext_mini_set_vskip_viii:`)

`__enumext_mini_addvspace_vii:`
`__enumext_mini_addvspace_viii:`

The functions `__enumext_mini_addvspace_vii:` and `__enumext_mini_addvspace_viii:` will apply the vertical space “*only above*” the `__enumext_mini_env*` environment on the *left side* when the `miniright` key is active in the `enumext*` and `keyans*` environments.

Here we will NOT take into account whether T_EX is in *horizontal mode* or *vertical mode*, since `\partopsep` is equal to 0pt in both environments.

```

1138 \cs_new_protected:Nn \__enumext_mini_addvspace_vii:

```

```

1139 {
1140   \__enumext_mini_set_vskip_vii:
1141   \par\nopagebreak
1142   \addvspace { \__enumext_minipage_left_skip }
1143 }
1144 \cs_new_protected:Nn \__enumext_mini_addvspace_viii:
1145 {
1146   \__enumext_mini_set_vskip_viii:
1147   \par\nopagebreak
1148   \addvspace { \__enumext_minipage_left_skip }
1149 }

```

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

10.19.4 The command `\miniright`

The command `\miniright` will close the `__enumext_mini_env*` environment on the “left side”, open the `__enumext_mini_env*` environment on the “right side” adding the *adjusted vertical space*. By default we will add `\centering` when starting the “right side” environment. The starred version ‘*’ inhibits the use of `\centering` command i.e. the usual L^AT_EX justification is maintained in the `__enumext_mini_env*` on the “right side”.

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

```

1150 \NewDocumentCommand \miniright { s }
1151 {
1152   \int_compare:nNnT { \__enumext_keyans_pic_level_int } = { 1 }
1153   {
1154     \msg_error:nnn { enumext } { wrong-miniright-place }
1155   }
1156   \int_compare:nNnT { \__enumext_level_int } = { 0 }
1157   {
1158     \msg_error:nnn { enumext } { wrong-miniright-place }
1159   }
1160   \int_compare:nNnTF { \__enumext_keyans_level_int } = { 1 }
1161   {
1162     \__enumext_keyans_mini_right_cmd:n {#1}
1163   }
1164   { \__enumext_mini_right_cmd:n {#1} }
1165 }

```

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

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

```

1166 \cs_new_protected:Npn \__enumext_mini_right_cmd:n #1
1167 {
1168   \dim_compare:nNnTF
1169   { \dim_use:c { \__enumext_minipage_right_ \__enumext_level: _dim } } > { \c_zero_dim }
1170   {
1171     \__enumext_multicols_stop:
1172     \end{\__enumext_mini_env*}
1173     \hfill
1174     \begin{\__enumext_mini_env*}
1175     { \dim_use:c { \__enumext_minipage_right_ \__enumext_level: _dim } }
1176     \par\addvspace { \__enumext_minipage_right_skip }
1177     \bool_if:nF {#1}
1178     {
1179       \centering
1180     }
1181     \int_gzero:N \g__enumext_minipage_stat_int
1182   }
1183   { \msg_error:nnn { enumext } { wrong-miniright-use } }
1184 }

```


(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 version* ‘`*`’ of the `\mini_right` 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.

```

1185 \cs_new_protected:Npn \__enumext_keyans_mini_right_cmd:n #1
1186 {
1187   \dim_compare:nNnTF { \__enumext_minipage_right_v_dim } > { \c_zero_dim }
1188   {
1189     \__enumext_keyans_multicols_stop:
1190     \end{__enumext_mini_env*}
1191     \hfill
1192     \begin{__enumext_mini_env*}{ \__enumext_minipage_right_v_dim }
1193     \par\addvspace { \__enumext_minipage_right_skip }
1194     \bool_if:nF {#1}
1195     {
1196       \centering
1197     }
1198     \int_gzero:N \g__enumext_minipage_stat_int
1199   }
1200   { \msg_error:nnn { enumext } { wrong-miniright-use } }
1201 }

```

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

10.20 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

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

above*

below

below*

```

1202 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1203 {
1204   \keys_define:nn { enumext / #1 }
1205   {
1206     above .skip_set:c = { \__enumext_vspace_above_#2_skip },
1207     above .value_required:n = true,
1208     above* .code:n      = \bool_set_true:c { \__enumext_vspace_a_star_#2_bool }
1209                      \keys_set:nn { enumext / #1 } { above = {##1} },
1210     above* .value_required:n = true,
1211     below .skip_set:c = { \__enumext_vspace_below_#2_skip },
1212     below .value_required:n = true,
1213     below* .code:n      = \bool_set_true:c { \__enumext_vspace_b_star_#2_bool }
1214                      \keys_set:nn { enumext / #1 } { below = {##1} },
1215     below* .value_required:n = true,
1216   }
1217 }
1218 \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }

```

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

10.20.1 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.

```

1219 \cs_new_protected:Nn \__enumext_vspace_above:
1220 {
1221   \skip_if_eq:nnF
1222   { \skip_use:c { \__enumext_vspace_above_ \__enumext_level: _skip } } { \c_zero_skip }
1223   {
1224     \bool_if:cTF { \__enumext_vspace_a_star_ \__enumext_level: _bool }
1225     {
1226       \vspace*{ \skip_use:c { \__enumext_vspace_above_ \__enumext_level: _skip } }
1227     }
1228     {
1229       \vspace { \skip_use:c { \__enumext_vspace_above_ \__enumext_level: _skip } }
1230     }
1231   }
1232 }

```

(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.

```

1233 \cs_new_protected:Nn \__enumext_vspace_below:
1234 {
1235   \skip_if_eq:nnF
1236   { \skip_use:c { \__enumext_vspace_below_ \__enumext_level: _skip } } { \c_zero_skip }
1237   {
1238     \bool_if:cTF { \__enumext_vspace_b_star_ \__enumext_level: _bool }
1239     {
1240       \vspace*{ \skip_use:c { \__enumext_vspace_below_ \__enumext_level: _skip } }
1241     }
1242     {
1243       \vspace { \skip_use:c { \__enumext_vspace_below_ \__enumext_level: _skip } }
1244     }
1245   }
1246 }

```

(End of definition for `__enumext_vspace_below:`.)

10.20.2 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.

```

1247 \cs_new_protected:Nn \__enumext_vspace_above_v:
1248 {
1249   \skip_if_eq:nnF { \__enumext_vspace_above_v_skip } { \c_zero_skip }
1250   {
1251     \bool_if:NTF \__enumext_vspace_a_star_v_bool
1252     {
1253       \vspace*{ \__enumext_vspace_above_v_skip }
1254     }
1255     { \vspace { \__enumext_vspace_above_v_skip } }
1256   }
1257 }

```

(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.

```

1258 \cs_new_protected:Nn \__enumext_vspace_below_v:
1259 {
1260   \skip_if_eq:nnF { \__enumext_vspace_below_v_skip } { \c_zero_skip }
1261   {
1262     \bool_if:NTF \__enumext_vspace_b_star_v_bool
1263     {
1264       \vspace*{ \__enumext_vspace_below_v_skip }
1265     }
1266     { \vspace { \__enumext_vspace_below_v_skip } }
1267   }
1268 }

```

(End of definition for `__enumext_vspace_below_v:`.)

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

`__enumext_vspace_above_vii:` and `__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.

`__enumext_vspace_above_viii:`

```

1269 \cs_new_protected:Nn \__enumext_vspace_above_vii:
1270 {
1271   \skip_if_eq:nnF { \__enumext_vspace_above_vii_skip } { \c_zero_skip }
1272   {
1273     \bool_if:NTF \__enumext_vspace_a_star_vii_bool
1274     {
1275       \vspace*{ \__enumext_vspace_above_vii_skip }
1276     }
1277     { \vspace { \__enumext_vspace_above_vii_skip } }
1278   }
1279 }
1280 \cs_new_protected:Nn \__enumext_vspace_above_viii:
1281 {
1282   \skip_if_eq:nnF { \__enumext_vspace_above_viii_skip } { \c_zero_skip }

```

```

1283     {
1284         \bool_if:NTF \l__enumext_vspace_a_star_viii_bool
1285         {
1286             \vspace*{ \l__enumext_vspace_above_viii_skip }
1287         }
1288         { \vspace { \l__enumext_vspace_above_viii_skip } }
1289     }
1290 }

```

(End of definition for \l__enumext_vspace_above_vii: and \l__enumext_vspace_above_viii:.)

The functions \l__enumext_vspace_below_vii: and \l__enumext_vspace_below_viii: apply the *vertical space below* the `enumext*` and `keyans*` environments set by the `below*` and `below` keys.

```

1291 \cs_new_protected:Nn \l__enumext_vspace_below_vii:
1292 {
1293     \skip_if_eq:nnF { \l__enumext_vspace_below_vii_skip } { \c_zero_skip }
1294     {
1295         \bool_if:NTF \l__enumext_vspace_b_star_vii_bool
1296         {
1297             \vspace*{ \l__enumext_vspace_below_vii_skip }
1298         }
1299         { \vspace { \l__enumext_vspace_below_vii_skip } }
1300     }
1301 }
1302 \cs_new_protected:Nn \l__enumext_vspace_below_viii:
1303 {
1304     \skip_if_eq:nnF { \l__enumext_vspace_below_viii_skip } { \c_zero_skip }
1305     {
1306         \bool_if:NTF \l__enumext_vspace_b_star_viii_bool
1307         {
1308             \vspace*{ \l__enumext_vspace_below_viii_skip }
1309         }
1310         { \vspace { \l__enumext_vspace_below_viii_skip } }
1311     }
1312 }

```

(End of definition for \l__enumext_vspace_below_vii: and \l__enumext_vspace_below_viii:.)

10.21 Setting save-ans and resume keys

The key `save-ans` is directly associated with the key `resume`, this will activate the entire “storage system” in the `enumext` package.

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

```

save-ans \keys_define:nn { enumext / level-1 }
resume  {
resume*  {
1315     save-ans .code:n = \l__enumext_storing_set:n {#1},
1316     save-ans .value_required:n = true,
1317     resume  .code:n = \l__enumext_resume_counter:,
1318     resume  .value_forbidden:n = true,
1319     resume* .code:n = \l__enumext_resume_counter_star:,
1320     resume* .value_forbidden:n = true,
1321 }
1322 \keys_define:nn { enumext / enumext* }
1323 {
1324     save-ans .code:n = \l__enumext_storing_set:n {#1},
1325     save-ans .value_required:n = true,
1326     resume  .code:n = \l__enumext_resume_counter_vii:,
1327     resume  .value_forbidden:n = true,
1328 }

```

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

The function `\l__enumext_storing_set:n` executed by the `save-ans` key sets the parameters for the operation of `\anskey`, `keyans` and `keyanspic`. The variable `\l__enumext_store_name_tl` will have the “store name” with which the *sequence* and *prop list* will be created.

The boolean var `\l__enumext_store_active_bool` will be set to true activating the entire internal *storage mechanism*, then the integer variable for the `resume` key will be created (if not exist), finally the function `\l__enumext_check_ans_int:n` will be called to activate the internal mechanism for checking the answers if the boolean variable `\l__enumext_check_ans_bool` set by `check-ans` key are active.

```

1329 \cs_new_protected:Npn \__enumext_storing_set:n #1
1330 {
1331   \tl_set:Nx \l__enumext_store_name_tl {#1}
1332   \bool_set_true:N \l__enumext_store_active_bool
1333   \int_if_exist:cF { g__enumext_resume_#1_int }
1334   {
1335     \int_new:c { g__enumext_resume_#1_int }
1336   }
1337   \bool_if:NT \l__enumext_check_ans_bool
1338   {
1339     \__enumext_check_ans_int:n {#1}
1340   }
1341 }

```

(End of definition for __enumext_storing_set:n.)

```

\__enumext_resume_counter:
\__enumext_resume_counter_vii:

```

The functions __enumext_resume_counter: and __enumext_resume_counter_vii: used by `resume` key in `enumext` and `enumext*`. If `save-ans` key present then set the start value from integer created by __enumext_storing_set:n.

```

1342 \cs_new_protected:Nn \__enumext_resume_counter:
1343 {
1344   \bool_if:NT \l__enumext_store_active_bool
1345   {
1346     \int_gset:Nn \g__enumext_resume_int
1347     {
1348       \int_use:c { g__enumext_resume_ \l__enumext_store_name_tl _int }
1349     }
1350   }
1351   \bool_set_true:N \l__enumext_resume_bool
1352 }
1353 \cs_new_protected:Nn \__enumext_resume_counter_vii:
1354 {
1355   \bool_if:NT \l__enumext_store_active_bool
1356   {
1357     \int_gset:Nn \g__enumext_resume_int
1358     {
1359       \int_use:c { g__enumext_resume_ \l__enumext_store_name_tl _int }
1360     }
1361   }
1362   \bool_set_true:N \l__enumext_resume_vii_bool
1363 }

```

(End of definition for __enumext_resume_counter: and __enumext_resume_counter_vii:.)

10.22 Setting check-ans key

The mechanism for checking that all questions are answered follows this logic:

If the line starts with an `\item` or `\item*` and does not open a nested environment it must have a `\anskey`, if the line starts with an `\item` or `\item*` and opens a *nested environment*, each `\item` or `\item*` in the *nested* environment must have a “once” `\anskey`.

In order for the mechanism for the check-answer to work (not counting `keyans` and `keyanspic`) we need:

1. We must keep track of the total number of `\item` and `\item*` that appear within the environment including the nested levels.
2. We must keep track of the total number of `\item` and `\item*` that appear per level of nesting.
3. Keeping track of the number of times the environment nests.

Each `\item` and `\item*` in the environment must be matched with the counter associated with `\anskey` (`\g__enumext_count_item_ans_int`). 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 increase by one (for the `\item` or `\item*` that opens the nest) so that the account remains the same.
- c) If there is the option `no-store` we must add the items within this level plus one to maintain the equality.

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.

10.22.1 The check answer mechanism

Now we define the keys `check-ans` and `no-store` for all levels of `enumext` and `enumext*` environments.

```

check-ans 1364 \cs_set_protected:Npn \__enumext_tmp:n #1
no-store 1365 {
1366   \keys_define:nn { enumext / #1 }
1367   {
1368     check-ans .bool_set:N = \__enumext_check_ans_bool,
1369     check-ans .initial:n = false,
1370     no-store .bool_set:N = \__enumext_store_ans_bool,
1371     no-store .initial:n = false,
1372   }
1373 }
1374 \clist_map_inline:nn
1375 {
1376   level-1, level-2, level-3, level-4, enumext*
1377 }
1378 { \__enumext_tmp:n {#1} }
```

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

The function `__enumext_check_ans_int:n` will create the integer variables for the internal checking answer mechanism used by the `check-ans` key. The integer variables take the form `\g__enumext_count_⟨store name⟩_item_ans_int` and `\g__enumext_count_⟨store name⟩_item_X_int`

```

1379 \cs_new_protected:Npn \__enumext_check_ans_int:n #1
1380 {
1381   \int_if_exist:cF { g__enumext_count_#1_item_ans_int }
1382   { \int_new:c { g__enumext_count_#1_item_ans_int } }
1383   \int_if_exist:cF { g__enumext_count_#1_i_int }
1384   { \int_new:c { g__enumext_count_#1_i_int } }
1385   \int_if_exist:cF { g__enumext_count_#1_ii_int }
1386   { \int_new:c { g__enumext_count_#1_ii_int } }
1387   \int_if_exist:cF { g__enumext_count_#1_iii_int }
1388   { \int_new:c { g__enumext_count_#1_iii_int } }
1389   \int_if_exist:cF { g__enumext_count_#1_iv_int }
1390   { \int_new:c { g__enumext_count_#1_iv_int } }
1391   \int_if_exist:cF { g__enumext_count_#1_vii_int }
1392   { \int_new:c { g__enumext_count_#1_vii_int } }
```

We make `\g__enumext_count_item_X_int` equal to the integer variable that contains all the occurrences of `\item` and `\item*` in the different levels and we will make `\g__enumext_count_item_ans_int` equal to the integer variable handled by the `\anskey` command.

```

1393   \bool_lazy_all:nTF
1394   {
1395     { \g__enumext_starred_bool }
1396     { \int_compare_p:nNn { \__enumext_level_int } = { \c_zero_int } }
1397   }
1398   {
1399     \int_gset_eq:Nc \g__enumext_count_item_all_int { g__enumext_count_#1_vii_int }
1400   }
1401   {
1402     \int_gset_eq:Nc \g__enumext_count_item_all_int { g__enumext_count_#1_i_int }
1403   }
1404   \int_gset_eq:Nc \g__enumext_count_item_i_int { g__enumext_count_#1_i_int }
1405   \int_gset_eq:Nc \g__enumext_count_item_ii_int { g__enumext_count_#1_ii_int }
1406   \int_gset_eq:Nc \g__enumext_count_item_iii_int { g__enumext_count_#1_iii_int }
1407   \int_gset_eq:Nc \g__enumext_count_item_iv_int { g__enumext_count_#1_iv_int }
1408   \int_gset_eq:Nc \g__enumext_count_item_vii_int { g__enumext_count_#1_vii_int }
1409   \int_gset_eq:Nc \g__enumext_count_item_ans_int { g__enumext_count_#1_item_ans_int }
1410 }
```

(End of definition for `__enumext_check_ans_int:n`.)

10.22.2 Set-up check answer mechanism

The function `__enumext_check_ans_count:` will count the number of times the `\item` and `\item*` commands appears per level within the `enumext` environment. The boolean variable `\l__enumext_store_ans_bool` controlled by the `no-store` key will increment the integer variable of the level counter by `1` to preserve the equality that we will use in the final comparison of the process.

```

1411 \cs_new_protected:Nn \__enumext_check_ans_count:
1412 {
1413   \bool_if:NT \l__enumext_check_ans_bool
```

```

1414     {
1415         \bool_if:NTF \l__enumext_store_ans_bool
1416         {
1417             \int_gadd:cn { g__enumext_count_item_ \__enumext_level: _int }
1418             { \int_use:c { g__enumext_count_level_ \__enumext_level: _int } + 1 }
1419         }
1420         { \int_gincr:c { g__enumext_count_item_ \__enumext_level: _int } }
1421     }
1422 }

```

(End of definition for __enumext_check_ans_count:.)

__enumext_check_ans_active: The function __enumext_check_ans_active: compare all \item's plus \item*'s and \item's with answer for checking answer mechanism and display the appropriate message on the terminal.

__enumext_check_ans_active_vii:

```

1423 \cs_new_protected:Nn \__enumext_check_ans_active:
1424 {
1425     \int_set:Nn \l__enumext_compare_items_ans_int
1426     {
1427         \g__enumext_count_item_all_int - \g__enumext_count_item_ii_int
1428         - \g__enumext_count_item_iii_int - \g__enumext_count_item_iv_int
1429     }
1430     \int_compare:nNnTF
1431     { \l__enumext_compare_items_ans_int } = { \g__enumext_count_item_ans_int }
1432     {
1433         \msg_term:nnV { enumext } { items-same-answer }
1434         \g__enumext_store_name_tl
1435     }
1436     {
1437         \msg_warning:nnV { enumext } { item-different-answer }
1438         \g__enumext_store_name_tl
1439     }

```

After the function is executed, we set the temporary integer variables to zero.

```

1440     \int_gzero:N \g__enumext_count_level_i_int
1441     \int_gzero:N \g__enumext_count_level_ii_int
1442     \int_gzero:N \g__enumext_count_level_iii_int
1443     \int_gzero:N \g__enumext_count_level_iv_int
1444     \int_gzero:N \g__enumext_count_level_vii_int
1445 }

```

```

1446 \cs_new_protected:Nn \__enumext_check_ans_active_vii:
1447 {
1448     \int_set:Nn \l__enumext_compare_items_ans_int
1449     {
1450         \g__enumext_count_item_all_int
1451         - \g__enumext_count_item_i_int
1452         - \g__enumext_count_item_ii_int
1453         - \g__enumext_count_item_iii_int
1454         - \g__enumext_count_item_iv_int
1455     }
1456     \int_compare:nNnTF
1457     { \l__enumext_compare_items_ans_int } = { \g__enumext_count_item_ans_int }
1458     {
1459         \msg_term:nnV { enumext } { items-same-answer }
1460         \g__enumext_store_name_tl
1461     }
1462     {
1463         \msg_warning:nnV { enumext } { item-different-answer }
1464         \g__enumext_store_name_tl
1465     }
1466     \int_gzero:N \g__enumext_count_level_i_int
1467     \int_gzero:N \g__enumext_count_level_ii_int
1468     \int_gzero:N \g__enumext_count_level_iii_int
1469     \int_gzero:N \g__enumext_count_level_iv_int
1470     \int_gzero:N \g__enumext_count_level_vii_int
1471 }

```

(End of definition for __enumext_check_ans_active: and __enumext_check_ans_active_vii:.)

10.23 Keys and functions associated with storage

We add the keys `wrap-ans`, `mark-ans`, `mark-pos`, `show-ans`, `show-pos`, `mark-ref` and `store-ref` related to the “*storage system*” and internal mechanism of “*label and ref*” only at the *first level* of `enumext` and `enumext*`.

```

wrap-ans 1472 \cs_set_protected:Npn \__enumext_tmp:n #1
mark-ans 1473 {
mark-pos 1474   \keys_define:nn { enumext / #1 }
store-ref 1475   {
1476     wrap-ans .cs_set_protected:Np = \__enumext_anskey_wrapper:n ##1,
1477     wrap-ans .initial:n = \fbox{##1},
1478     wrap-ans .value_required:n = true,
1479     mark-ans .code:n = \tl_set:Nn \l__enumext_mark_answer_sym_tl {##1},
1480     mark-ans .initial:n = \textasteriskcentered,
1481     mark-ans .value_required:n = true,
1482     mark-pos .choice:,
1483     mark-pos / left .code:n = \str_set:Nn \l__enumext_mark_position_str { l },
1484     mark-pos / right .code:n = \str_set:Nn \l__enumext_mark_position_str { r },
1485     mark-pos .initial:n = right,
1486     mark-pos .value_required:n = true,
1487     show-ans .code:n = \bool_set_true:N \l__enumext_show_answer_bool
1488               \bool_set_false:N \l__enumext_show_position_bool,
1489     show-ans .value_forbidden:n = true,
1490     show-pos .code:n = \bool_set_true:N \l__enumext_show_position_bool
1491               \bool_set_false:N \l__enumext_show_answer_bool,
1492     show-pos .value_forbidden:n = true,
1493     mark-ref .code:n = \tl_set:Nn \l__enumext_mark_ref_sym_tl {##1},
1494     mark-ref .initial:n = \textasteriskcentered,
1495     mark-ref .value_required:n = true,
1496     store-ref .bool_set:N = \l__enumext_store_ref_key_bool,
1497     store-ref .initial:n = false,
1498   }
1499 }
1500 \clist_map_inline:nn { level-1, enumext* } { \__enumext_tmp:n {#1} }

```

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

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

```

mark-pos 1501 \cs_set_protected:Npn \__enumext_tmp:n #1
show-ans 1502 {
1503   \keys_define:nn { enumext / #1 }
1504   {
1505     mark-pos .choice:,
1506     mark-pos / left .code:n = \str_set:Nn \l__enumext_mark_position_str { l },
1507     mark-pos / right .code:n = \str_set:Nn \l__enumext_mark_position_str { r },
1508     mark-pos .initial:n = right,
1509     mark-pos .value_required:n = true,
1510     show-ans .code:n = \bool_set_true:N \l__enumext_show_answer_bool
1511                   \bool_set_false:N \l__enumext_show_position_bool,
1512     show-ans .value_forbidden:n = true,
1513     show-pos .code:n = \bool_set_true:N \l__enumext_show_position_bool
1514                   \bool_set_false:N \l__enumext_show_answer_bool,
1515     show-pos .value_forbidden:n = true,
1516   }
1517 }
1518 \clist_map_inline:nn { keyans, keyans* } { \__enumext_tmp:n {#1} }

```

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

For the `enumext` and `enumext*` environments we will only add the keys `columns*` and `columns-sep*`. The values set by these keys will be passed as optional arguments to the “*inner levels*” of the `enumext` and `enumext*` environments via the `__enumext_store_level_open:` function used by the “*storage system*” to preserve the structure and then used by the `\printkeyans` command.

```

1519 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1520 {
1521   \keys_define:nn { enumext / #1 }
1522   {
1523     columns* .code:n = \bool_set_true:c { l__enumext_store_columns_#2_bool }
1524                   \int_set:cn { l__enumext_store_columns_#2_int } {##1}
1525                   \tl_put_right:ce { l__enumext_store_opt_#2_tl }

```



```

1526         {
1527             columns = \exp_not:v { l__enumext_store_columns_#2_int },
1528         },,
1529         columns* .value_required:n = true,
1530         columns-sep* .code:n = \bool_set_true:c { l__enumext_store_columns_sep_#2_bool }
1531             \dim_set:cn { l__enumext_store_columns_sep_#2_dim } {##1}
1532             \tl_put_right:ce { l__enumext_store_opt_#2_tl }
1533             {
1534                 columns-sep = \exp_not:v { l__enumext_store_columns_sep_#2_dim },
1535             },
1536         columns-sep* .value_required:n = true,
1537     }
1538 }
1539 \clist_map_inline:nn
1540 {
1541     {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {enumext*}{vii}
1542 }
1543 { \__enumext_tmp:nn #1 }

```

(End of definition for `columns*` and `columns-sep*`.)

10.23.1 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, if it does not exist it will create it globally. The “stored content” is retrieved by means of the `\getkeyans` command.

The form in which the content is “stored” in the *prop list* is $\{\langle position \rangle\}\{\langle content \rangle\}$. This function is used by `\anskey` in `enumext` and `enumext*` environments, `\item*` in `keyans` and `keyans*` environments and `\anspic` in `keyanspic` environment.

```

1544 \cs_new_protected:Npn \__enumext_store_addto_prop:n #1
1545 {
1546     \prop_if_exist:cF { g__enumext_ \l__enumext_store_name_tl _prop }
1547     { \prop_new:c { g__enumext_ \l__enumext_store_name_tl _prop } }
1548     \prop_gput:cen { g__enumext_ \l__enumext_store_name_tl _prop }
1549     {
1550         \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } +1 }
1551     } { #1 }
1552 }
1553 \cs_generate_variant:Nn \__enumext_store_addto_prop:n { V }

```

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

10.23.2 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, if it does not exist it will create it globally. 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.

```

1554 \cs_new_protected:Npn \__enumext_store_addto_seq:n #1
1555 {
1556     \seq_if_exist:cF { g__enumext_ \l__enumext_store_name_tl _seq }
1557     { \seq_new:c { g__enumext_ \l__enumext_store_name_tl _seq } }
1558     \seq_gput_right:cn { g__enumext_ \l__enumext_store_name_tl _seq } { #1 }
1559 }
1560 \cs_generate_variant:Nn \__enumext_store_addto_seq:n { v, V }

```

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

10.23.3 Functions for storing the list structure in the sequence

```

\__enumext_store_level_open:
\__enumext_store_level_close:

```

The memorization structure of the list is handled by the functions `__enumext_store_level_open:` and `__enumext_store_level_close:` which are executed per level within the `enumext` environment. As this structure will be stored in the sequence set by the `save-ans` key, we will not be able to modify it locally, so it is better to take only two copies of the values set by the `columns` and `columns-sep` keys if they are present when changing levels within the `enumext` environment when executing `\anskey`. We will store these values in the variable `\l__enumext_store_columns_X_tl` if they are different from `0` and `opt` and pass them as an optional argument to the environment stored in the sequence `enumext`.

```

1561 \cs_new_protected:Nn \__enumext_store_level_open:
1562 {
1563     \bool_if:NF \l__enumext_store_ans_bool
1564     {

```

```

1565 \tl_if_empty:cTF { l__enumext_store_opt_ \__enumext_level: _tl }
1566 {
1567     \__enumext_store_addto_seq:n
1568     {
1569         \item \begin{enumext}
1570     }
1571 }
1572 {
1573     \tl_put_left:cn { l__enumext_store_opt_ \__enumext_level: _tl }
1574     {
1575         \item \begin{enumext} [
1576     }
1577     \tl_put_right:cn { l__enumext_store_opt_ \__enumext_level: _tl }
1578     {
1579         ]
1580     }
1581     \__enumext_store_addto_seq:v { l__enumext_store_opt_ \__enumext_level: _tl }
1582 }
1583 }
1584 }
1585 \cs_new_protected:Nn \__enumext_store_level_close:
1586 {
1587     \bool_if:NF \l__enumext_store_ans_bool
1588     {
1589         \__enumext_store_addto_seq:n { \end{enumext} }
1590     }
1591 }

```

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

__enumext_store_level_open_vii:
 __enumext_store_level_close_vii:

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

```

1592 \cs_new_protected:Nn \__enumext_store_level_open_vii:
1593 {
1594     \bool_if:NF \l__enumext_store_ans_bool
1595     {
1596         \tl_if_empty:NTF \l__enumext_store_opt_vii_tl
1597         {
1598             \__enumext_store_addto_seq:n
1599             {
1600                 \item \mode_leave_vertical:
1601                 \vspace { -\skip_eval:n { \baselineskip + \parsep } }
1602                 \begin{enumext*}[before={\setlength{\topsep}{\opt}},]
1603             }
1604         }
1605         {
1606             \tl_put_left:Nn \l__enumext_store_opt_vii_tl
1607             {
1608                 \item \mode_leave_vertical:
1609                 \vspace { -\skip_eval:n { \baselineskip + \parsep } }
1610                 \begin{enumext*}[before={\setlength{\topsep}{\opt}},
1611             }
1612             \tl_put_right:Nn \l__enumext_store_opt_vii_tl
1613             {
1614                 ]
1615             }
1616             \__enumext_store_addto_seq:v \l__enumext_store_opt_vii_tl
1617         }
1618     }
1619 }
1620 \cs_new_protected:Nn \__enumext_store_level_close_vii:
1621 {
1622     \bool_if:NF \l__enumext_store_ans_bool
1623     {
1624         \__enumext_store_addto_seq:n { \end{enumext*} }
1625     }
1626 }

```

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

10.23.4 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

1627 \cs_new_protected:Nn \__enumext_print_keyans_box:NN
1628 {
1629   \mode_leave_vertical:
1630   \skip_horizontal:n { -\dim_use:N #2 }
1631   \makebox[0pt][ r ]
1632   {
1633     \makebox[ \dim_use:N #1 ][ \l__enumext_mark_position_str ]
1634     {
1635       \tl_use:N \l__enumext_mark_answer_sym_tl
1636     }
1637   }
1638   \skip_horizontal:n { \dim_use:N #2 }
1639 }
1640 \cs_generate_variant:Nn \__enumext_print_keyans_box:NN { cc }
```

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

10.24 The command `\anskey` and internal label and ref

Since we will be “*storing content*” in a list environment within *(sequences)* and can (more or less) manage the options passed to each level, it is necessary that we have a little more control over `\item` when storing. The `\anskey` command will cover this point and give it very similar behaviour to that of `\item` in the `enumext` and `enumext*` environments.

`\anskey` We want the command to be executed as follows: `\anskey(<number>)*[<key = val>]{<content>}` so first we’ll add the keys `item-sym*`, `item-pos*` and `store-brk`.

```
1641 \keys_define:nn { enumext / anskey }
1642 {
1643   item-sym* .tl_set:N = \l__enumext_store_item_symbol_tl,
1644   item-sym* .value_required:n = true,
1645   item-pos* .dim_set:N = \l__enumext_store_item_symbol_sep_dim,
1646   item-pos* .value_required:n = true,
1647   store-brk .bool_set:N = \l__enumext_store_columns_break_bool,
1648   store-brk .default:n = true,
1649   store-brk .value_forbidden:n = true,
1650 }
```

This command `\anskey` will only be present when using the `save-ans` key in `enumext` and `enumext*` environments, otherwise it will return an error. If the `check-ans` key is active, increment `\g__enumext-count_item_ans_int`, then call internal function `__enumext_store_anskey_code:nnnn` will “*store content*” in the *(sequence)* and in the *(prop list)*.

```
1651 \NewDocumentCommand \anskey { d() s o +m }
1652 {
1653   \bool_if:NF \l__enumext_store_active_bool
1654   {
1655     \msg_error:nnnn { enumext } { anskey-wrong-place }{ anskey }{ enumext }
1656   }
1657   \int_compare:nNnT { \l__enumext_keyans_level_int } = { 1 }
1658   {
1659     \msg_error:nnnn { enumext } { command-wrong-place }{ anskey }{ keyans }
1660   }
1661   \int_compare:nNnT { \l__enumext_keyans_pic_level_int } = { 1 }
1662   {
1663     \msg_error:nnnn { enumext } { command-wrong-place }{ anskey }{ keyanspic }
1664   }
1665   \group_begin:
1666     \bool_if:NF \l__enumext_store_ans_bool
1667     {
1668       \bool_if:NT \l__enumext_check_ans_bool
1669       {
1670         \int_gincr:N \g__enumext_count_item_ans_int
1671       }
1672       \__enumext_store_anskey_code:nnnn {#1} {#2} {#3} {#4}
1673     }
1674 }
```

```

1674   \group_end:
1675   }

```

(End of definition for `\anskey`. This function is documented on page 10.)

```
\__enumext_store_anskey_code:nnnn
```

The internal function `__enumext_store_anskey_code:nnnn` first we pass the command `<argument>` to the `<prop list>`, then checks the state of the variable `\l__enumext_store_ref_key_bool` handled by the `store-ref` key and will call the function `\l__enumext_store_internal_ref:` for the internal “label and ref” system. Followed by this if the `show-ans` or `show-pos` keys are active we will show the “wrapped” `<argument>` passed to the command.

```

1676 \cs_new_protected:Npn \__enumext_store_anskey_code:nnnn #1 #2 #3 #4
1677 {
1678   \__enumext_store_addto_prop:n {#4}
1679   \bool_if:NT \l__enumext_store_ref_key_bool
1680   {
1681     \__enumext_store_internal_ref:
1682   }
1683   \__enumext_store_anskey_show_left:n { #4 }

```

Now we start processing the optional arguments 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 `[<key = val>]`, if the `store-brk` key is present and the command is running under `enumext` (not in the starred version) we will add `\columnbreak` and then `\item`.

```

1684   \tl_clear:N \l__enumext_store_anskey_arg_tl
1685   \tl_if_novalue:nF {#3}
1686   {
1687     \keys_set:nn { enumext / anskey } {#3}
1688   }
1689   \bool_lazy_and:nnT
1690   { \l__enumext_store_columns_break_bool }
1691   { \bool_not_p:n { \l__enumext_starred_bool } }
1692   {
1693     \tl_put_left:Nn \l__enumext_store_anskey_arg_tl { \columnbreak }
1694   }
1695   \tl_put_right:Nn \l__enumext_store_anskey_arg_tl { \item }

```

Now we will check the `<(number)>` argument and add it to `\l__enumext_store_anskey_arg_tl` if the command is running under `enumext*` (starred version).

```

1696   \tl_if_novalue:nF {#1}
1697   {
1698     \int_set:Nn \l__enumext_store_columns_join_int {#1}
1699     \bool_if:NT \l__enumext_starred_bool
1700     {
1701       \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
1702       {
1703         ( \exp_not:V \l__enumext_store_columns_join_int )
1704       }
1705     }
1706   }

```

And now we will review the starred argument `*` together with the keys `item-sym*` and `item-pos*` and pass them to `\l__enumext_store_anskey_arg_tl`.

```

1707   \bool_if:nTF {#2}
1708   {
1709     \tl_put_right:Nn \l__enumext_store_anskey_arg_tl { * }
1710     \tl_if_empty:NF \l__enumext_store_item_symbol_tl
1711     {
1712       \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
1713       {
1714         [ \exp_not:V \l__enumext_store_item_symbol_tl ]
1715       }
1716     }
1717     \dim_compare:nT
1718     {
1719       \l__enumext_store_item_symbol_sep_dim != \c_zero_dim
1720     }
1721     {
1722       \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
1723       {
1724         [ \exp_not:V \l__enumext_store_item_symbol_sep_dim ]

```

```

1725         }
1726     }
1727     \tl_put_right:Nn \l__enumext_store_anskey_arg_tl {#4}
1728 }
1729 {
1730     \tl_put_right:Nn \l__enumext_store_anskey_arg_tl {#4}
1731 }

```

Finally we check if the `store-ref` key is active along with the `hyperref` package load, if both conditions are met, it will create the `\hyperlink` and then store in `\sequence`.

```

1732 \bool_lazy_and:nnT
1733 { \l__enumext_store_ref_key_bool }
1734 { \l__enumext_hyperref_bool }
1735 {
1736     \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
1737     {
1738         \hfill \exp_not:N \hyperlink { \exp_not:V \l__enumext_newlabel_arg_one_tl }
1739         { \exp_not:V \l__enumext_mark_ref_sym_tl }
1740     }
1741 }
1742 \__enumext_store_addto_seq:V \l__enumext_store_anskey_arg_tl
1743 }

```

(End of definition for `__enumext_store_anskey_code:nnnn`.)

`__enumext_store_internal_ref:`

The function `__enumext_store_internal_ref:` handles the internal “*label and ref*” system used by the `store-ref` and `mark-ref` keys for `\anskey` will allow to execute `\ref{<store name>:position}` and will return `1.(a).i.A`.

First we will remove the dots “.” from the *current* `<labels>`, we do not want to get double dots in our references, then we will place this in the variable `\l__enumext_newlabel_arg_two_tl`.

```

1744 \cs_new_protected:Nn \__enumext_store_internal_ref:
1745 {
1746     \cs_set_protected:Npn \__enumext_tmp:n ##1
1747     {
1748         \tl_set_eq:cc { \l__enumext_label_copy_##1_tl } { \l__enumext_label_##1_tl }
1749         \tl_reverse:c { \l__enumext_label_copy_##1_tl }
1750         \tl_remove_once:cn { \l__enumext_label_copy_##1_tl } { . }
1751         \tl_reverse:c { \l__enumext_label_copy_##1_tl }
1752     }
1753     \clist_map_inline:nn { i, ii, iii, iv, vii } { \__enumext_tmp:n {##1} }
1754     \cs_set:Npn \__enumext_tmp:n ##1
1755     { . \tl_use:c { \l__enumext_label_copy_ \int_to_roman:n {##1} _tl } }

```

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

```

1756 \bool_lazy_all:nT
1757 {
1758     { \g__enumext_starred_bool }
1759     { \int_compare_p:nNn { \l__enumext_level_int } = { \c_zero_int } }
1760 }
1761 {
1762     \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
1763     { \tl_use:N \l__enumext_label_copy_vii_tl }
1764 }
1765 \bool_lazy_all:nT
1766 {
1767     { \l__enumext_standar_bool }
1768     { \g__enumext_starred_bool }
1769     { \int_compare_p:nNn { \l__enumext_level_int } > { \c_zero_int } }
1770 }
1771 {
1772     \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
1773     {
1774         \tl_use:N \l__enumext_label_copy_vii_tl
1775         \int_step_function:nnN { 1 } { { \l__enumext_level_int } \__enumext_tmp:n
1776     }
1777 }

```

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

```

1778 \bool_lazy_all:nT
1779 {

```

```

1780     { \l__enumext_standar_bool }
1781     { \int_compare_p:nNn { \l__enumext_level_int } > { \c_zero_int } }
1782     { \int_compare_p:nNn { \l__enumext_level_h_int } = { \c_zero_int } }
1783     { \bool_not_p:n { \l__enumext_starred_bool } }
1784   }
1785   {
1786     \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
1787     {
1788       \tl_use:N \l__enumext_label_copy_i_tl
1789       \int_step_function:nnN { 2 } { \l__enumext_level_int } \__enumext_tmp:n
1790     }
1791   }
1792   \cs_set:Npn \__enumext_tmp:n ##1
1793   { \tl_use:c { \l__enumext_label_copy_ \int_to_roman:n {##1} _tl } }
1794   \bool_lazy_all:nT
1795   {
1796     { \l__enumext_standar_bool }
1797     { \int_compare_p:nNn { \l__enumext_level_int } > { \c_zero_int } }
1798     { \bool_not_p:n { \g__enumext_starred_bool } }
1799     { \int_compare_p:nNn { \l__enumext_level_h_int } > { \c_zero_int } }
1800   }
1801   {
1802     \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
1803     {
1804       \int_step_function:nnN { 1 } { \l__enumext_level_int } \__enumext_tmp:n
1805       . \tl_use:N \l__enumext_label_copy_vii_tl
1806     }
1807   }

```

Now we set the variable `\l__enumext_newlabel_arg_one_tl` which will contain $\langle \textit{store name} : \textit{position} \rangle$.

```

1808   \tl_put_right:Ne \l__enumext_newlabel_arg_one_tl
1809   {
1810     \l__enumext_store_name_tl \c_colon_str
1811     \int_eval:n { \prop_count:c { \g__enumext_ \l__enumext_store_name_tl _prop } }
1812   }

```

Now execute the function `__enumext_newlabel:nn` and save the result in the variable `\l__enumext_store_write_aux_file_tl` and finally we write in the `.aux` file.

```

1813   \tl_put_right:Ne \l__enumext_store_write_aux_file_tl
1814   {
1815     \__enumext_newlabel:nn
1816     { \exp_not:V \l__enumext_newlabel_arg_one_tl }
1817     { \l__enumext_newlabel_arg_two_tl }
1818   }
1819   \l__enumext_store_write_aux_file_tl
1820 }

```

(End of definition for `__enumext_store_internal_ref:.`)

`__enumext_store_anskey_show_wrap:n`

The function `__enumext_store_anskey_show_wrap:n` “wraps” the $\langle \textit{argument} \rangle$ passed to `\anskey` when using the `wrap-ans` key.

```

1821 \cs_new_protected:Npn \__enumext_store_anskey_show_wrap:n #1
1822 {
1823   \par
1824   \bool_if:NT \l__enumext_starred_bool
1825   {
1826     \cs_set:Nn \__enumext_level: { vii }
1827   }
1828   \__enumext_print_keyans_box:cc
1829   { \l__enumext_labelwidth_ \__enumext_level: _dim }
1830   { \l__enumext_labelsep_ \__enumext_level: _dim }
1831   \__enumext_anskey_wrapper:n { #1 }
1832 }

```

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

`__enumext_store_anskey_show_left:n`

The function `__enumext_store_anskey_show_left:n` will show the “mark” defined by the `mark-ans` key or the “position” of the content stored in the $\langle \textit{prop list} \rangle$ when using the `show-pos` key on the left margin next to the “wraps” $\langle \textit{argument} \rangle$ passed to `\anskey` on the right side when using the `show-ans` key.

```

1833 \cs_new_protected:Npn \__enumext_store_anskey_show_left:n #1
1834 {
1835   \bool_if:NT \l__enumext_show_answer_bool
1836   {
1837     \__enumext_store_anskey_show_wrap:n { #1 }
1838   }
1839   \bool_if:NT \l__enumext_show_position_bool
1840   {
1841     \tl_set:Nx \l__enumext_mark_answer_sym_tl
1842     {
1843       \group_begin:
1844       \exp_not:N \normalfont
1845       \exp_not:N \footnotesize [ \int_eval:n
1846       {
1847         \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
1848       }
1849       ]
1850       \group_end:
1851     }
1852     \__enumext_store_anskey_show_wrap:n { #1 }
1853   }
1854 }

```

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

10.25 Common functions for keyans and keyanspic

10.25.1 Storing content in prop list

`__enumext_keyans_addto_prop:n`

The function `__enumext_keyans_addto_prop:n` will pass the contents of the `\l__enumext_label_v_tl` (current *label*) for the `keyans` environment and the `\l__enumext_label_vi_tl` (current *label*) 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_keyans_label_tl` variable, which will be passed to the *prop list* defined by the `save-ans` key using the `__enumext_store_addto_prop:V`.

```

1855 \cs_new_protected:Npn \__enumext_keyans_addto_prop:n #1
1856 {
1857   \tl_clear:N \l__enumext_store_keyans_label_tl
1858   \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
1859   {
1860     \tl_put_right:Nx \l__enumext_store_keyans_label_tl { \l__enumext_label_vi_tl }
1861   }
1862   {
1863     \tl_put_right:Nx \l__enumext_store_keyans_label_tl { \l__enumext_label_v_tl }
1864   }
1865   \tl_if_novalue:nF { #1 }
1866   {
1867     \tl_put_right:Nx \l__enumext_store_keyans_label_tl { \c_space_tl #1 }
1868   }
1869   \__enumext_store_addto_prop:V \l__enumext_store_keyans_label_tl
1870 }

```

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

10.25.2 The store-ref key for keyans and keyanspic

The internal “*label and ref*” system for the `keyans` and `keyanspic` environments has slight differences with the one implemented for the `\anskey` command, basically because in both environments we are interested in the *current label*.

`__enumext_keyans_internal_ref:`

The function `__enumext_keyans_internal_ref:` handles the internal “*label and ref*” system used by the `store-ref` and `mark-ref` keys for `\item*` and `\anspic*` commands. The mechanism defined here will allow to execute `\ref{<store name>:position}` and will return 1. (A).

```

1871 \cs_new_protected:Npn \__enumext_keyans_internal_ref:
1872 {

```

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`.

```

1873   \cs_set_protected:Npn \__enumext_tmp:n ##1
1874   {
1875     \tl_set_eq:cc { \l__enumext_label_copy_##1_tl } { \l__enumext_label_##1_tl }
1876     \tl_reverse:c { \l__enumext_label_copy_##1_tl }
1877     \tl_remove_once:cn { \l__enumext_label_copy_##1_tl } { . }

```



```

1878     \tl_reverse:c { l__enumext_label_copy_##1_tl }
1879   }
1880   \clist_map_inline:nn { i, v, vi, vii, viii } { \__enumext_tmp:n {##1} }
1881   \int_compare:nNnTF { \__enumext_keyans_pic_level_int } = { 1 }
1882   {
1883     \tl_put_right:Ne \__enumext_newlabel_arg_two_tl
1884       { \__enumext_label_copy_i_tl . \__enumext_label_copy_vi_tl }
1885   }
1886   {
1887     \tl_put_right:Ne \__enumext_newlabel_arg_two_tl
1888       { \__enumext_label_copy_i_tl . \__enumext_label_copy_v_tl }
1889   }

```

Now we set the variable `__enumext_newlabel_arg_one_tl` which will contain $\langle \textit{store name} : \textit{position} \rangle$.

```

1890   \tl_put_right:Ne \__enumext_newlabel_arg_one_tl
1891   {
1892     \__enumext_store_name_tl \c_colon_str
1893     \int_eval:n { \prop_count:c { g__enumext_ \__enumext_store_name_tl _prop } }
1894   }

```

Now execute the function `__enumext_newlabel:nn` and save the result in the variable `__enumext_store_write_aux_file_tl` and finally we write in the `.aux` file.

```

1895   \tl_put_right:Ne \__enumext_store_write_aux_file_tl
1896   {
1897     \__enumext_newlabel:nn
1898       { \exp_not:V \__enumext_newlabel_arg_one_tl }
1899       { \__enumext_newlabel_arg_two_tl }
1900   }
1901   \bool_if:NT \__enumext_store_ref_key_bool
1902   {
1903     \__enumext_store_write_aux_file_tl
1904   }
1905 }

```

(End of definition for `__enumext_keyans_internal_ref:.`)

10.25.3 Storing content in sequence

`__enumext_keyans_addto_seq:n`

The function `__enumext_keyans_addto_seq:n` will pass the contents of the `__enumext_label_v_tl` (“current label”) for the `keyans` environment and the `__enumext_label_vi_tl` (current label) for the `keyanspic` environment when using `\item*` and `\anspic*`, followed by the contents of the optional argument of both commands to the `__enumext_store_keyans_label_tl` variable to the sequence defined by the `save-ans` key.

```

1906 \cs_new_protected:Npn \__enumext_keyans_addto_seq:n #1
1907 {
1908   \tl_clear:N \__enumext_store_keyans_label_tl
1909   \int_compare:nNnTF { \__enumext_keyans_pic_level_int } = { 1 }
1910   {
1911     \tl_put_right:Ne \__enumext_store_keyans_label_tl { \item \__enumext_label_vi_tl }
1912   }
1913   {
1914     \tl_put_right:Ne \__enumext_store_keyans_label_tl { \item \__enumext_label_v_tl }
1915   }
1916   \tl_if_novalue:nF { #1 }
1917   {
1918     \tl_put_right:Ne \__enumext_store_keyans_label_tl { \c_space_tl #1 }
1919   }

```

Checks if the `store-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.

```

1920   \bool_lazy_and:nnT
1921     { \__enumext_store_ref_key_bool }
1922     { \__enumext_hyperref_bool }
1923   {
1924     \tl_put_right:Ne \__enumext_store_keyans_label_tl
1925     {
1926       \hfill \exp_not:N \hyperlink
1927       {
1928         \exp_not:V \__enumext_newlabel_arg_one_tl
1929       }
1930       { \exp_not:V \__enumext_mark_ref_sym_tl }
1931     }

```

```

1932     }
1933     \__enumext_store_addto_seq:V \l__enumext_store_keyans_label_tl

```

Finally, copy the contents of the variable `\l__enumext_store_keyans_label_tl` into the global variable `\g__enumext_check_ans_item_tl` to be used by the function `__enumext_keyans_check_ans:nn` and increment the value of the integer variable `\g__enumext_count_item_ans_int` handled by the `check-ans` key.

```

1934     \tl_gset:NV \g__enumext_check_ans_item_tl \l__enumext_store_keyans_label_tl
1935     \bool_if:NT \l__enumext_check_ans_bool
1936     {
1937         \int_gincr:N \g__enumext_count_item_ans_int
1938     }
1939 }

```

(End of definition for `__enumext_keyans_addto_seq:n`.)

10.25.4 Check for starred commands

`__enumext_keyans_check_ans:nn`

The function `__enumext_keyans_check_ans:nn` performs an extra check for the `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.

```

1940 \cs_new_protected:Npn \__enumext_keyans_check_ans:nn #1 #2
1941 {
1942     \tl_if_empty:NTF \g__enumext_check_ans_item_tl
1943     {
1944         \msg_warning:nnnn { enumext } { missing-starred }{ #1 }{ #2 }
1945     }
1946     { \tl_gclear:N \g__enumext_check_ans_item_tl }
1947 }

```

(End of definition for `__enumext_keyans_check_ans:nn`.)

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

The code is very similar to the `\anskey` code, but, if I change the order of the operations the counter off label are incorrect.

`__enumext_keyans_show_left:n`

Common function to show *starred commands* and *(position)* of stored content in *(prop list)* for `keyans` and `keyanspic`. Need add `1` to `\l__enumext_mark_answer_sym_tl` for `keyans` environment.

```

1948 \cs_new_protected:Npn \__enumext_keyans_show_left:n #1
1949 {
1950     \bool_if:NT \l__enumext_show_answer_bool
1951     {
1952         \tl_put_left:Nn \l__enumext_label_v_tl
1953         {
1954             \__enumext_print_keyans_box:NN
1955             \l__enumext_labelwidth_i_dim
1956             \l__enumext_labelsep_i_dim
1957         }
1958         \tl_if_novalue:nF { #1 }
1959         { \tl_put_right:Nn \l__enumext_label_v_tl { \c_space_tl [ #1 ] } }
1960     }
1961     \bool_if:NT \l__enumext_show_position_bool
1962     {
1963         \tl_set:Nc \l__enumext_mark_answer_sym_tl
1964         {
1965             \group_begin:
1966             \exp_not:N \normalfont
1967             \exp_not:N \footnotesize [ \int_eval:n
1968             {
1969                 \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
1970                 + \l__enumext_keyans_level_int
1971             }
1972             ]
1973             \group_end:
1974         }
1975         \tl_put_left:Nn \l__enumext_label_v_tl
1976         {
1977             \__enumext_print_keyans_box:NN
1978             \l__enumext_labelwidth_i_dim
1979             \l__enumext_labelsep_i_dim
1980         }
1981         \tl_if_novalue:nF { #1 }

```

```

1982         { \tl_put_right:Nn \__enumext_label_v_tl { \c_space_tl [ #1 ] } }
1983     }
1984 }

```

(End of definition for `__enumext_keyans_show_left:n`.)

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

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

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

```

1985 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1986 {
1987     \keys_define:nn { enumext / #1 }
1988     {
1989         item-sym* .tl_set:c = { \__enumext_item_symbol_#2_tl },
1990         item-sym* .value_required:n = true,
1991         item-sym* .initial:n = { $\star$ },
1992         item-pos* .dim_set:c = { \__enumext_item_symbol_sep_#2_dim },
1993         item-pos* .value_required:n = true,
1994     }
1995 }
1996 \clist_map_inline:nn
1997 {
1998     {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {enumext*}{vii}
1999 }
2000 { \__enumext_tmp:nn #1 }

```

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

10.27 Redefining `\footnote` command

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

```

2001 \cs_new_protected:Nn \__enumext_footnotetext:nn
2002 {
2003     \footnotetext[#1]{#2}
2004 }
2005 \cs_new_protected:Nn \__enumext_renew_footnote:
2006 {
2007     \seq_gclear:N \g__enumext_footnote_arg_seq
2008     \seq_gclear:N \g__enumext_footnote_int_seq
2009     \RenewDocumentCommand \footnote { o +m }
2010     {
2011         \tl_if_novalue:nTF {##1}
2012         {
2013             \stepcounter{footnote}
2014             \int_gset_eq:Nc \g__enumext_footnote_int { c@footnote }
2015         }
2016         {
2017             \int_gset:Nn \g__enumext_footnote_int { ##1 }
2018         }
2019         \footnotemark [ \g__enumext_footnote_int ]
2020         \seq_gput_right:Nn \g__enumext_footnote_arg_seq { ##2 }
2021         \seq_gput_right:NV \g__enumext_footnote_int_seq \g__enumext_footnote_int
2022     }
2023 }
2024 \cs_new_protected:Nn \__enumext_print_footnote:
2025 {
2026     \seq_if_empty:NF \g__enumext_footnote_int_seq
2027     {
2028         \seq_map_pairwise_function:NNN
2029         \g__enumext_footnote_int_seq
2030         \g__enumext_footnote_arg_seq
2031         \__enumext_footnotetext:nn
2032     }
2033 }

```

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

10.28 Redefining \item command

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

10.28.1 The \item command in enumext

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

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

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

The boolean variable `\l__enumext_wrap_label_X_bool` is used by the function `__enumext_make_label: (§10.29)`.

```

2034 \cs_new_protected:Npn \__enumext_default_item:n
2035 {
2036   \tl_if_novalue:nTF {#1}
2037   {
2038     \bool_if:NT \l__enumext_check_ans_bool
2039     {
2040       \int_gincr:N \g__enumext_count_item_all_int
2041       \int_gincr:c { g__enumext_count_level_ \__enumext_level: _int }
2042     }
2043     \bool_set_true:c { l__enumext_wrap_label_ \__enumext_level: _bool }
2044     \__enumext_item_std:w \tl_use:c { l__enumext_fake_item_indent_ \__enumext_level: _tl }
2045   }
2046   {
2047     \bool_set_eq:cc
2048     { l__enumext_wrap_label_ \__enumext_level: _bool }
2049     { l__enumext_wrap_label_opt_ \__enumext_level: _bool }
2050     \__enumext_item_std:w [#1] \tl_use:c { l__enumext_fake_item_indent_ \__enumext_level: _tl }
2051   }
2052 }

```

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

The `\item*`, `\item*[⟨symbol⟩]` and `\item*[⟨symbol⟩][⟨offset⟩]` works like the numbered `\item`, but placing a `[⟨symbol⟩]` to the “left” of the `⟨label⟩` separated from it by the value set by the `labelsep` key and can be *offset* using the second optional argument `[⟨offset⟩]`.

#1: `\l__enumext_item_symbol_X_tl`

#2: `\l__enumext_item_symbol_sep_X_dim`

First we will make a copy of `\l__enumext_item_symbol_X_tl` which is set by the key `item-sym*` or passed as optional argument in the global variable `\g__enumext_item_symbol_tl`, followed by setting the variable `\l__enumext_item_symbol_sep_X_dim` set by the key `item*-sep` or by the second optional argument.

Then we will see the state of the variable `\l__enumext_check_ans_bool` set by the key `check-ans`, set the boolean variable `\l__enumext_wrap_label_X_bool` to “true” and execute `__enumext_item_std:w`.

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

The boolean variable `\l__enumext_wrap_label_X_bool` and the vars `\l__enumext_item_symbol_sep_X_dim`, `\g__enumext_item_symbol_tl` are used by the function `__enumext_make_label: (§10.29)`.

```

2053 \cs_new_protected:Npn \__enumext_starred_item:nn #1 #2
2054 {
2055   \tl_if_novalue:nF {#1}
2056   {
2057     \tl_set:cn { l__enumext_item_symbol_ \__enumext_level: _tl } {#1}
2058   }
2059   \tl_gset_eq:Nc \g__enumext_item_symbol_tl { l__enumext_item_symbol_ \__enumext_level: _tl }
2060   \tl_if_novalue:nTF {#2}
2061   {
2062     \dim_set_eq:cc
2063     { l__enumext_item_symbol_sep_ \__enumext_level: _dim }
2064     { l__enumext_labelsep_ \__enumext_level: _dim }
2065   }

```

```

2066     {
2067         \dim_set:cn { l__enumext_item_symbol_sep_ \__enumext_level: _dim } {#2}
2068     }
2069     \bool_if:NT \l__enumext_check_ans_bool
2070     {
2071         \int_gincr:N \g__enumext_count_item_all_int
2072         \int_gincr:c { g__enumext_count_level_ \__enumext_level: _int }
2073     }
2074     \bool_set_true:c { l__enumext_wrap_label_ \__enumext_level: _bool }
2075     \__enumext_item_std:w \tl_use:c { l__enumext_fake_item_indent_ \__enumext_level: _tl }
2076 }

```

(End of definition for __enumext_starred_item:nn.)

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

This function is passed to `__enumext_list_arg_two_X:` which is used in the definition of the `enumext` environment (§10.31).

```

2077 \cs_new_protected:Nn \__enumext_redefine_item:
2078 {
2079     \RenewDocumentCommand \item { s o o }
2080     {
2081         \bool_if:nTF {##1}
2082         {
2083             \__enumext_starred_item:nn {##2} {##3}
2084         }
2085         { \__enumext_default_item:n {##2} }
2086     }
2087 }

```

(End of definition for __enumext_redefine_item:.)

10.28.2 The `\item` command in keyans

The `\item*` and `\item*[\langle content \rangle]` commands *store* the current *label* next to the `[\langle content \rangle]` 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`.

```

2088 \cs_new_protected:Npn \__enumext_keyans_default_item:n #1
2089 {
2090     \tl_if_novalue:nTF { #1 }
2091     {
2092         \bool_set_true:N \l__enumext_wrap_label_v_bool
2093         \__enumext_item_std:w \tl_use:N \l__enumext_fake_item_indent_v_tl
2094     }
2095     {
2096         \bool_set_eq:NN \l__enumext_wrap_label_v_bool \l__enumext_wrap_label_opt_v_bool
2097         \__enumext_item_std:w [#1] \tl_use:N \l__enumext_fake_item_indent_v_tl
2098     }
2099 }

```

(End of definition for __enumext_keyans_default_item:n.)

`__enumext_keyans_starred_item:n` The function `__enumext_keyans_starred_item:n` which will make a temporary copy of the “current label”, execute the `show-ans` or `show-pos` keys using the function `__enumext_keyans_show_left:n` and will display the contents of that item using the internal copy `__enumext_item_std:w`, this is necessary to prevent incrementing the current “counter” of the original *label*.

```

2100 \cs_new_protected:Npn \__enumext_keyans_starred_item:n #1
2101 {
2102     \tl_set_eq:NN \l__enumext_keyans_tmpa_tl \l__enumext_label_v_tl
2103     \__enumext_keyans_show_left:n { #1 }
2104     \bool_set_true:N \l__enumext_wrap_label_v_bool
2105     \__enumext_item_std:w \tl_use:N \l__enumext_fake_item_indent_v_tl

```

Recover the original value of the “current label” and *store* it first in the *prop list* (including the optional argument), run the internal “label and ref” system if the `store-ref` key is active and finally *store* it in the *sequence*.

```

2106     \tl_set_eq:NN \l__enumext_label_v_tl \l__enumext_keyans_tmpa_tl
2107     \__enumext_keyans_addto_prop:n { #1 }
2108     \__enumext_keyans_internal_ref:
2109     \__enumext_keyans_addto_seq:n { #1 }
2110 }

```

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

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

This function is passed to `__enumext_list_arg_two_v:` which is used in the definition of the `keyans` environment (§10.31).

```
2111 \cs_new_protected:Nn \__enumext_keyans_redefine_item:
2112 {
2113   \RenewDocumentCommand \item { s o }
2114   {
2115     \bool_if:nTF {##1}
2116     {
2117       \peek_remove_spaces:n
2118       {
2119         \__enumext_keyans_starred_item:n {##2}
2120       }
2121     }
2122     {
2123       \__enumext_keyans_default_item:n {##2}
2124     }
2125   }
2126 }
```

(End of definition for `\item*` and `__enumext_keyans_redefine_item:`. This function is documented on page 11.)

10.29 Redefining `\makeLabel` command

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

10.29.1 Redefining `\makeLabel` for `enumext`

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

```
2127 \cs_new_protected:Nn \__enumext_item_starred:
2128 {
2129   \tl_if_empty:cF { \__enumext_item_symbol_ \__enumext_level: _tl }
2130   {
2131     \mode_leave_vertical:
2132     \skip_horizontal:n { -\dim_use:c { \__enumext_item_symbol_sep_ \__enumext_level: _dim } }
2133     \makebox[0pt][r]{ \tl_use:N \g__enumext_item_symbol_tl }
2134     \skip_horizontal:n { \dim_use:c { \__enumext_item_symbol_sep_ \__enumext_level: _dim } }
2135   }
2136 }
```

(End of definition for `__enumext_item_starred:`.)

`__enumext_make_label:` The function `__enumext_make_label:` redefine `\makeLabel` for the `enumext` environment.

This function is passed to `__enumext_list_arg_two_X:` which is used in the definition of the `enumext` environment (§10.31).

```
2137 \cs_new_protected:Nn \__enumext_make_label:
2138 {
2139   \RenewDocumentCommand \makeLabel { m }
2140   {
2141     \tl_use:c { \__enumext_label_fill_left_ \__enumext_level: _tl }
2142     \tl_use:c { \__enumext_label_font_style_ \__enumext_level: _tl }
2143     \bool_if:cTF { \__enumext_wrap_label_ \__enumext_level: _bool }
2144     {
2145       \__enumext_item_starred:
2146       \use:c { __enumext_wrapper_label_ \__enumext_level: :n } { ##1 }
2147     }
2148     { ##1 }
2149     \tl_use:c { \__enumext_label_fill_right_ \__enumext_level: _tl }
2150     \tl_gclear:N \g__enumext_item_symbol_tl
2151   }
2152 }
```

(End of definition for `__enumext_make_label:`.)

10.29.2 Redefining `\makeLabel` for `keyans`

`__enumext_keyans_make_label:` The function `__enumext_keyans_make_label:` redefine `\makeLabel` for `keyans` environment. This function is passed to `__enumext_list_arg_two_v:` which is used in the definition of the `keyans` environment (§10.31).

```

2153 \cs_new_protected:Nn \__enumext_keyans_make_label:
2154 {
2155   \RenewDocumentCommand \makeLabel { m }
2156   {
2157     \tl_use:N \l__enumext_label_fill_left_v_tl
2158     \tl_use:N \l__enumext_label_font_style_v_tl
2159     \bool_if:NTF \l__enumext_wrap_label_v_bool
2160     {
2161       \__enumext_wrapper_label_v:n { ##1 }
2162     }
2163     { ##1 }
2164     \tl_use:N \l__enumext_label_fill_right_v_tl
2165   }
2166 }

```

(End of definition for `__enumext_keyans_make_label:`)

10.30 Calculation of `\leftmargin` and `\itemindent`

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

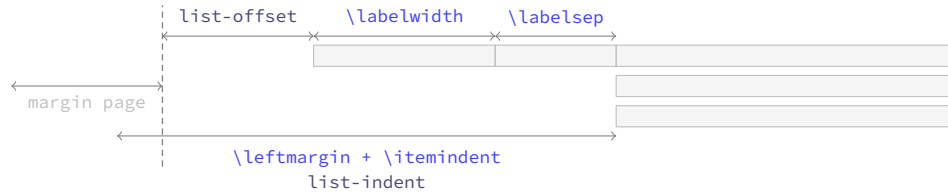


Figure 9: Representation of standard horizontal lengths in `list` environment.

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

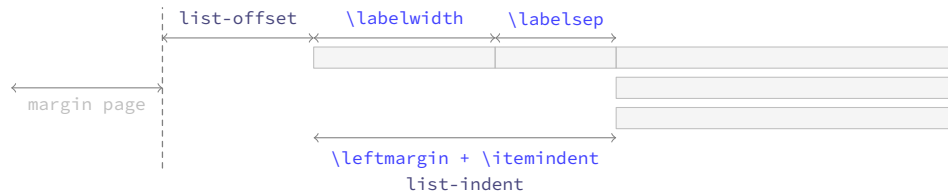


Figure 10: Representation of horizontal lengths concept in list in `enumext`.

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

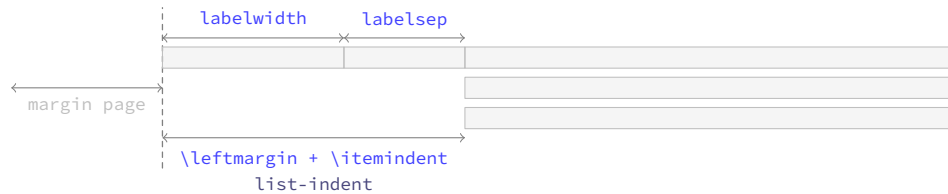


Figure 11: Default horizontal lengths in `enumext`.

```

\__enumext_calc_hspace:NNNNNN
\__enumext_calc_hspace:cccccc

```

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

```

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

```

And returns the “adjusted” values of `\leftmargin` and `\itemindent`.

This function is passed to `__enumext_list_arg_two_X:` which is used in the definition of the `enumext` and `keyans` environments (§10.31).

```

2167 \cs_new_protected:Npn \__enumext_calc_hspace:NNNNNN #1#2#3#4#5#6#7
2168 {
2169   \dim_compare:nNnT { #1 } < { \c_zero_dim }

```



```

2170     {
2171       \msg_warning:nnnV { enumext } { width-non-positive }{ labelwidth }{ #1 }
2172       \dim_set:Nn #1 { \dim_abs:n { #1 } }
2173     }
2174     \dim_compare:nNnT { #2 } < { \c_zero_dim }
2175     {
2176       \msg_warning:nnnV { enumext } { width-negative }{ labelsep }{ #2 }
2177       \dim_set:Nn #2 { \dim_abs:n { #2 } }
2178     }

```

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

```

2179     \bool_if:nF #7 { \dim_set:Nn #4 { #1 + #2 } }

```

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

```

2180     \dim_compare:nNnTF { #4 } < { \c_zero_dim }
2181     {
2182       \dim_set:Nn #6 { #1 + #2 - #4 }
2183       \dim_set:Nn #5 { #1 + #2 + #3 - #6 }
2184     }
2185     {
2186       \dim_compare:nNnT { #4 } = { #1 + #2 }
2187       { \dim_set:Nn #6 { \c_zero_dim } }
2188       \dim_compare:nNnT { #4 } < { #1 + #2 }
2189       { \dim_set:Nn #6 { #1 + #2 - #4 } }
2190       \dim_compare:nNnT { #4 } > { #1 + #2 }
2191       {
2192         \dim_set:Nn #6 { -#1 - #2 + #4 }
2193         \dim_set:Nn #6 { #6*-1 }
2194       }
2195       \dim_set:Nn #5 { #1 + #2 + #3 - #6 }
2196     }
2197   }
2198   \cs_generate_variant:Nn \__enumext_calc_hspace:NNNNNNN { ccccccc }

```

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

10.31 Setting second argument of the lists

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

In this function for the second list argument we will implement the keys `start`, `resume` and `show-length` together with the redefinition of `\item` for `enumext` and `keyans` environments. 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.

```

2199   \cs_set_protected:Npn \__enumext_tmp:n #1
2200   {
2201     \cs_new_protected:cpn { __enumext_list_arg_two_#1: }
2202     {
2203       \__enumext_calc_hspace:cccccc
2204       { \__enumext_labelwidth_#1_dim } { \__enumext_labelsep_#1_dim }
2205       { \__enumext_listoffset_#1_dim } { \__enumext_leftmargin_tmp_#1_dim }
2206       { \__enumext_leftmargin_#1_dim } { \__enumext_itemindent_#1_dim }
2207       { \__enumext_leftmargin_tmp_#1_bool }
2208       \clist_map_inline:nn
2209       { labelsep, labelwidth, itemindent, leftmargin, rightmargin, listparindent }
2210       { \dim_set_eq:cc {####1} { \__enumext_####1_#1_dim } }
2211       \clist_map_inline:nn { topsep, parsep, partopsep, itemsep }
2212       { \skip_set_eq:cc {####1} { \__enumext_####1_#1_skip } }
2213       \usecounter { enumX#1 }
2214       \bool_lazy_and:nnTF
2215       { \str_if_eq_p:nn {#1} { i } }
2216       { \bool_if_p:N \__enumext_resume_bool }
2217       { \setcounter { enumXi } { \int_eval:n { \g__enumext_resume_int } } }
2218       {
2219         \setcounter { enumX#1 }
2220         { \int_eval:n { \int_use:c { \__enumext_start_#1_int } - 1 } }
2221       }
2222       \str_if_eq:nnTF {#1} { v }

```

```

2223     {
2224         \__enumext_keyans_redefine_item:
2225         \__enumext_keyans_make_label:
2226         \__enumext_keyans_fake_item:
2227         \bool_if:cT { \__enumext_show_length_#1_bool }
2228         {
2229             \msg_term:nnnn { enumext } { list-lengths-not-nested } { v } { keyans }
2230         }
2231     }
2232     {
2233         \__enumext_redefine_item:
2234         \__enumext_make_label:
2235         \__enumext_use_key_ref:
2236         \__enumext_fake_item:
2237         \bool_if:cT { \__enumext_show_length_#1_bool }
2238         {
2239             \msg_term:nnne { enumext } { list-lengths } {#1} { \int_use:N \__enumext_level_#1 }
2240         }
2241     }
2242 }
2243 }
2244 \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 `\opt`. 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.

```

2245 \cs_set_protected:Npn \__enumext_tmp:n #1
2246 {
2247     \cs_new_protected:cpn { __enumext_list_arg_two_#1: }
2248     {
2249         \__enumext_calc_hspace:ccccc
2250         { \__enumext_labelwidth_#1_dim } { \__enumext_labelsep_#1_dim }
2251         { \__enumext_listoffset_#1_dim } { \__enumext_leftmargin_tmp_#1_dim }
2252         { \__enumext_leftmargin_#1_dim } { \__enumext_itemindent_#1_dim }
2253         { \__enumext_leftmargin_tmp_#1_bool }
2254         \clist_map_inline:nn
2255         { labelsep, labelwidth, itemindent, leftmargin, rightmargin, listparindent }
2256         { \dim_set_eq:cc {####1} { \__enumext_####1_#1_dim } }
2257         \clist_map_inline:nn { topsep, parsep, partopsep, itemsep }
2258         { \skip_set_eq:cc {####1} { \__enumext_####1_#1_skip } }
2259         \skip_set_eq:Nc \parsep { \__enumext_itemsep_#1_skip }
2260         \skip_zero:N \partopsep
2261         \usecounter { enumX#1 }
2262         \bool_lazy_and:nnTF
2263         { \str_if_eq_p:nn {#1} { vii } } { \bool_if_p:N \__enumext_resume_vii_bool }
2264         { \setcounter { enumXvii } { \int_eval:n { \g__enumext_resume_vii_int } } }
2265         {
2266             \setcounter { enumX#1 }
2267             { \int_eval:n { \int_use:c { \__enumext_start_#1_int } - 1 } }
2268         }
2269         \__enumext_use_key_ref_h:
2270         \str_if_eq:nnTF {#1} { vii }
2271         {
2272             \__enumext_fake_item_vii:
2273             \bool_if:cT { \__enumext_show_length_vii_bool }
2274             { \msg_term:nnnn { enumext } { list-lengths-not-nested } { vii } { enumext* } }
2275         }
2276         {
2277             \__enumext_fake_item_viii:
2278             \bool_if:cT { \__enumext_show_length_#1_bool }
2279             { \msg_term:nnnn { enumext } { list-lengths-not-nested } { #1 } { keyans* } }
2280         }
2281     }
2282 }
2283 \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:.)

10.32 The environment enumext

`enumext` We create the `enumext` environment based on `list` environment by levels and the internal `__enumext_mini_env*` environment.

```

2284 \NewDocumentEnvironment{enumext}{ 0{} }
2285 {
2286   \__enumext_safe_exec:
2287   \__enumext_parse_keys:n {#1}
2288   \__enumext_before_list:
2289   \__enumext_start_store_level:
2290   \__enumext_start_list:nn
2291   { \tl_use:c { l__enumext_label_ \__enumext_level: _tl } }
2292   {
2293     \use:c { __enumext_list_arg_two_ \__enumext_level: : }
2294     \__enumext_before_keys_exec:
2295   }
2296   \__enumext_after_args_exec:
2297 }
2298 {
2299   \__enumext_stop_list:
2300   \__enumext_stop_store_level:
2301   \__enumext_after_list:
2302 }

```

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

`__enumext_safe_exec:` First check the maximum nesting level for the `enumext` environment.

```

2303 \cs_new_protected:Nn \__enumext_safe_exec:
2304 {
2305   \int_incr:N \l__enumext_level_int
2306   \int_compare:nNnT { \l__enumext_level_int } > { 4 }
2307   { \msg_fatal:nn { enumext } { list-too-deep } }
2308   \bool_set_true:N \l__enumext_standar_bool
2309   \bool_lazy_all:nT
2310   {
2311     { \bool_not_p:n { \l__enumext_starred_bool } }
2312     { \int_compare_p:nNn { \l__enumext_level_h_int } = { \c_zero_int } }
2313   }
2314   {
2315     \bool_gset_true:N \g__enumext_standar_bool
2316   }
2317 }

```

(End of definition for `__enumext_safe_exec:`)

`__enumext_parse_keys:n` Parse [`key = val`] by levels in `enumext`. If the variable `\l__enumext_store_active_bool` is true it will call the function `__enumext_parse_store_keys:n` and reprocess the keys to pass them to the storage sequence.

```

2318 \cs_new_protected:Npn \__enumext_parse_keys:n #1
2319 {
2320   \exp_args:Ne \keys_set:nn
2321   { enumext / level-\int_use:N \l__enumext_level_int } {#1}
2322   \bool_if:NT \l__enumext_store_active_bool
2323   {
2324     \__enumext_parse_store_keys:n {#1}
2325   }
2326 }

```

(End of definition for `__enumext_parse_keys:n`)

`__enumext_parse_store_keys:n` The function `__enumext_parse_store_keys:n` searches for the values of the `columns` and `columns-sep` keys in the optional arguments per-level in `enumext` environment as long as the starred versions of the `columns*` and `columns-sep*` keys are not active. The captured values are stored in the variable `\l__enumext_store_opt_X_tl` which is used by the function `__enumext_store_level_open:`.

```

2327 \cs_new_protected:Npn \__enumext_parse_store_keys:n #1
2328 {
2329   \bool_if:cF { l__enumext_store_columns_ \__enumext_level: _bool }
2330   {
2331     \regex_match:nnT { \b columns\b } {#1}
2332     {
2333       \int_set_eq:cc

```

```

2334         { l__enumext_store_columns_ \__enumext_level: _int }
2335         { l__enumext_columns_ \__enumext_level: _int }
2336         \tl_put_right:ce { l__enumext_store_opt_ \__enumext_level: _tl }
2337         {
2338             columns = \exp_not:v { l__enumext_store_columns_ \__enumext_level: _int },
2339         }
2340     }
2341 }
2342 \bool_if:cF { l__enumext_store_columns_sep_ \__enumext_level: _bool }
2343 {
2344     \regex_match:nnT { \b columns-sep \b} {#1}
2345     {
2346         \dim_set_eq:cc
2347         { l__enumext_store_columns_sep_ \__enumext_level: _dim }
2348         { l__enumext_columns_sep_ \__enumext_level: _dim }
2349         \tl_put_right:ce { l__enumext_store_opt_ \__enumext_level: _tl }
2350         {
2351             columns-sep = \exp_not:v { l__enumext_store_columns_sep_ \__enumext_level: _dim }
2352         }
2353     }
2354 }
2355 }

```

(End of definition for __enumext_parse_store_keys:n.)

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

```

2356 \cs_new_protected:Nn \__enumext_start_store_level:
2357 {
2358     \bool_lazy_and:nnT { \__enumext_store_active_bool } { \bool_not_p:n { \__enumext_keyans_env: } }
2359     {
2360         \int_compare:nNnT { \__enumext_level_h_int } = { 1 }
2361         {
2362             \bool_set_true:c { l__enumext_store_upper_level_ \__enumext_level: _bool }
2363             \__enumext_store_level_open:
2364         }
2365         \int_compare:nNnT { \__enumext_level_int } > { 1 }
2366         {
2367             \bool_set_true:c { l__enumext_store_upper_level_ \__enumext_level: _bool }
2368             \__enumext_store_level_open:
2369         }
2370     }
2371 }
2372 \cs_new_protected:Nn \__enumext_stop_store_level:
2373 {
2374     \bool_if:cT { l__enumext_store_upper_level_ \__enumext_level: _bool }
2375     {
2376         \__enumext_store_level_close:
2377     }
2378 }

```

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

__enumext_before_list: The function __enumext_before_list: will add the vertical spacing on the environment if the above key is active next to the {⟨code⟩} defined by the before* key if it is active.

```

2379 \cs_new_protected:Nn \__enumext_before_list:
2380 {
2381     \__enumext_vspace_above:
2382     \__enumext_before_args_exec:

```

The function __enumext_check_ans_count: will handle the check answer mechanism, which will be activated with the check-ans key.

```

2383 \__enumext_check_ans_count:

```

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

```

2384 \dim_compare:nNtT
2385 { \dim_use:c { l__enumext_minipage_right_ \__enumext_level: _dim } } > { \c_zero_dim }
2386 {
2387   \dim_set:cn { l__enumext_minipage_left_ \__enumext_level: _dim }
2388   {
2389     \linewidth
2390     - \dim_use:c { l__enumext_minipage_right_ \__enumext_level: _dim }
2391     - \dim_use:c { l__enumext_minipage_hsep_ \__enumext_level: _dim }
2392   }

```

The boolean variable `\l__enumext_minipage_active_X_bool` will be activated and the integer variable `\g__enumext_minipage_stat_int` used by the `\miniright` command will be incremented, then the function `__enumext_mini_addvspace:` is called and the `__enumext_mini_env*` environment on the “left side” will be initialized followed by the “vertical spacing” applied to preserve the “baseline” between the *left* and *right* side environments. After these actions, the function `__enumext_multicols_start:` is called to handle the `multicols` environment.

- Here we use the plain T_EX macro `\nointerlineskip` to prevent baseline “glue” being added between the next pair of boxes in a *vertical list*.

```

2393   \bool_set_true:c { l__enumext_minipage_active_ \__enumext_level: _bool }
2394   \int_gincr:N \g__enumext_minipage_stat_int
2395   \__enumext_mini_addvspace:
2396   \nointerlineskip\noindent
2397   \begin{\__enumext_mini_env*}
2398   { \dim_use:c { l__enumext_minipage_left_ \__enumext_level: _dim } }
2399   }
2400   \__enumext_multicols_start:
2401   }

```

(End of definition for `__enumext_before_list:`)

`__enumext_multicols_start:`

The function `__enumext_multicols_start:` will start the `multicols` environment according to the value passed by the `columns` key, then set the default value for `\columnsep` when `columns-sep=opt` and set the value of `\multicolsep` equal to zero and leave `\columnseprule` equal to zero for inner levels.

```

2402 \cs_new_protected:Nn \__enumext_multicols_start:
2403 {
2404   \int_compare:nNtT
2405   { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } > { 1 }
2406   {
2407     \dim_compare:nNtT
2408     { \dim_use:c { l__enumext_columns_sep_ \__enumext_level: _dim } } = { \c_zero_dim }
2409     {
2410       \dim_set:cn { l__enumext_columns_sep_ \__enumext_level: _dim }
2411       {
2412         ( \dim_use:c { l__enumext_labelwidth_ \__enumext_level: _dim }
2413         + \dim_use:c { l__enumext_labelsep_ \__enumext_level: _dim }
2414         ) / \int_use:c { l__enumext_columns_ \__enumext_level: _int }
2415         - \dim_use:c { l__enumext_listoffset_ \__enumext_level: _dim }
2416       }
2417     }
2418     \dim_set_eq:Nc \columnsep { l__enumext_columns_sep_ \__enumext_level: _dim }
2419     \skip_zero:N \multicolsep
2420     \int_compare:nNtT { \l__enumext_level_int } > { 1 }
2421     {
2422       \dim_zero:N \columnseprule
2423     }

```

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.

```

2424   \bool_if:cF { l__enumext_minipage_active_ \__enumext_level: _bool }
2425   {
2426     \__enumext_multi_addvspace:
2427   }
2428   \raggedcolumns
2429   \begin{multicols}{ \int_use:c { l__enumext_columns_ \__enumext_level: _int } }
2430   }
2431   }

```

(End of definition for `__enumext_multicols_start:`)

`__enumext_multicols_stop:` The function `__enumext_multicols_stop:` will stop the `multicols` environment. If the boolean variable `\l__enumext_minipage_active_X_bool` is false (not nested in `__enumext_mini_env*`) we will apply our “vertical adjust” spacing.

```

2432 \cs_new_protected:Nn \__enumext_multicols_stop:
2433 {
2434   \int_compare:nNt
2435     { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } > { 1 }
2436   {
2437     \end{multicols}
2438     \bool_if:cF { l__enumext_minipage_active_ \__enumext_level: _bool }
2439     {
2440       \par\addvspace{ \skip_use:c { l__enumext_multicols_below_ \__enumext_level: _skip } }
2441     }
2442   }

```

If the `check-ans` key is active, we set the boolean variable `\g__enumext_check_ans_show_bool` to true and copy the stored name to the variable `\g__enumext_store_name_tl`. These variables will be used by the function `__enumext_after_env:n` to display the result of the internal check answer mechanism in the terminal.

```

2443   \bool_lazy_and:nnT { \l__enumext_check_ans_bool }{ \bool_not_p:n { \g__enumext_starred_bool }
2444   {
2445     \bool_gset_true:N \g__enumext_check_ans_show_bool
2446     \tl_gset:NV \g__enumext_store_name_tl \l__enumext_store_name_tl
2447   }
2448 }

```

(End of definition for `__enumext_multicols_stop:`.)

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

```

2449 \cs_new_protected:Nn \__enumext_after_list:
2450 {
2451   \bool_if:cTF { l__enumext_minipage_active_ \__enumext_level: _bool }
2452   {
2453     \int_compare:nNt { \g__enumext_minipage_stat_int } = { 1 }
2454     {
2455       \msg_warning:nn { enumext } { missing-miniright }
2456       \miniright
2457     }
2458     \int_gzero:N \g__enumext_minipage_stat_int
2459     \end{__enumext_mini_env*}
2460     \par\addvspace { \l__enumext_minipage_after_skip }
2461   }
2462   { \__enumext_multicols_stop: }

```

Now apply the `{\code}` handled by the `after` key together with the *vertical space* handled by the `below` key if they are present.

```

2463   \__enumext_after_stop_list:
2464   \__enumext_vspace_below:

```

Finally save the *current value* of the counter in `\g__enumext_resume_int` for the `resume` key. If the `save-ans` key is active, it will create the integer variable for the `resume` key, we only have to assign it the value of the current counter.

```

2465   \bool_set_false:N \l__enumext_standar_bool
2466   \int_gset_eq:NN \g__enumext_resume_int \value{enumXi}
2467   \int_if_exist:cT { g__enumext_resume_ \l__enumext_store_name_tl _int }
2468   {
2469     \int_gset_eq:cN
2470     { g__enumext_resume_ \l__enumext_store_name_tl _int }
2471     { \value{enumXi} }
2472   }
2473 }

```

(End of definition for `__enumext_after_list:`.)

As we don’t want our check to be executed `check-ans` by levels but on the complete list, we will take it out of the `enumext` environment using the “hook” function `__enumext_after_env:nn`.

```

2474 \__enumext_after_env:nn {enumext}
2475 {

```

```

2476 \bool_if:NT \g__enumext_check_ans_show_bool
2477 {
2478   \int_compare:nNnT { \l__enumext_level_int } = { 0 }
2479   {
2480     \__enumext_check_ans_active:
2481   }
2482 }
2483 \bool_gset_false:N \g__enumext_check_ans_show_bool
2484 \tl_gclear:N \g__enumext_store_name_tl
2485 }

```

10.33 The environment keyans

The environment `keyans` also based on lists. The main differences with the `enumext` environment are the *nesting* and the way the *answers* (choice) will be stored and checked, this environment is intended exclusively for “multiple choice questions”.

`keyans` Now we define the environment `keyans` also based on lists.

```

2486 \NewDocumentEnvironment{keyans}{ 0{ } }
2487 {
2488   \__enumext_keyans_safe_exec:
2489   \__enumext_keyans_parse_keys:n {#1}
2490   \__enumext_before_list_v:
2491   \__enumext_start_list:nn
2492   { \tl_use:N \l__enumext_label_v_tl }
2493   {
2494     \__enumext_list_arg_two_v:
2495     \__enumext_before_keys_exec_v:
2496   }
2497   \__enumext_after_args_exec_v:
2498 }
2499 {
2500   \__enumext_keyans_check_ans:nn { item }{ keyans }
2501   \__enumext_stop_list:
2502   \__enumext_after_list_v:
2503 }

```

(End of definition for `keyans`. This function is documented on page 11.)

`__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.

```

2504 \cs_new_protected:Nn \__enumext_keyans_safe_exec:
2505 {
2506   \bool_if:NF \l__enumext_store_active_bool
2507   {
2508     \msg_error:nnnn { enumext } { wrong-place }{ keyans }{ save-ans }
2509   }
2510   \int_incr:N \l__enumext_keyans_level_int
2511   \bool_set_true:N \l__enumext_keyans_env_bool
2512   % Set false for interfering with enumext nested in keyans (yes, its possible and crayze)
2513   \bool_set_false:N \l__enumext_store_active_bool
2514   \int_compare:nNnT { \l__enumext_keyans_level_int } > { 1 }
2515   {
2516     \msg_error:nn { enumext } { keyans-nested }
2517   }
2518   \int_compare:nNnT { \l__enumext_level_int } > { 1 }
2519   {
2520     \msg_error:nn { enumext } { keyans-wrong-level }
2521   }
2522 }

```

(End of definition for `__enumext_keyans_safe_exec:.`)

`__enumext_keyans_parse_keys:n` Parse [`key = val`] for `keyans` environment.

```

2523 \cs_new_protected:Npn \__enumext_keyans_parse_keys:n #1
2524 {
2525   \keys_set:nn { enumext / keyans } {#1}
2526 }

```

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

`__enumext_before_list_v:` The function `__enumext_before_list_v:` will add the *vertical spacing above* the environment if the `above` key is active next to the *(code)* defined by the `before` key if it is active.

```

2527 \cs_new_protected:Nn \__enumext_before_list_v:
2528 {
2529     \__enumext_vspace_above_v:
2530     \__enumext_before_args_exec_v:

```

When the `mini-env` key is active it will set the value of the `\l__enumext_minipage_right_v_dim` to be the *width* of the `__enumext_mini_env*` environment on the *left side*, using this value together with the value of the `\l__enumext_minipage_hsep_v_dim` set by the `mini-sep` key, the value of `\l__enumext_minipage_left_v_dim` will be set, which will be the *width* of `__enumextt_mini_env*` environment on the *right side*, always having `\linewidth` as the maximum width between them.

```

2531     \dim_compare:nNnT { \l__enumext_minipage_right_v_dim } > { \c_zero_dim }
2532     {
2533         \dim_set:Nn \l__enumext_minipage_left_v_dim
2534         {
2535             \linewidth - \l__enumext_minipage_right_v_dim - \l__enumext_minipage_hsep_v_dim
2536         }

```

The boolean variable `\l__enumext_minipage_active_v_bool` will be activated and the integer variable `\g__enumext_minipage_stat_int` used by the `\miniright` command will be incremented, then the function `__enumext_keyans_mini_addvspace:` is called and the `__enumext_mini_env*` environment on *left side* will be initialized followed by the *vertical spacing* `\l__enumext_minipage_left_skip`. Here we use the plain TeX macro `\nointerlineskip` to prevent baseline “glue” being added between the next pair of boxes in a *vertical list*.

```

2537         \bool_set_true:N \l__enumext_minipage_active_v_bool
2538         \int_gincr:N \g__enumext_minipage_stat_int
2539         \__enumext_keyans_mini_addvspace:
2540         \nointerlineskip\noindent
2541         \begin{\__enumext_mini_env*}{ \l__enumext_minipage_left_v_dim }
2542     }

```

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

```

2543     \__enumext_keyans_multicols_start:
2544 }

```

(End of definition for `__enumext_before_list_v:`)

`__enumext_keyans_multicols_start:` The function `__enumext_keyans_multicols_start:` will start the `multicols` environment according to the value passed by the `columns` key.

```

2545 \cs_new_protected:Nn \__enumext_keyans_multicols_start:
2546 {
2547     \int_compare:nNnT { \l__enumext_columns_v_int } > { 1 }
2548     {

```

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

```

2549         \dim_compare:nNnT { \l__enumext_columns_sep_v_dim } = { \c_zero_dim }
2550         {
2551             \dim_set:Nn \l__enumext_columns_sep_v_dim
2552             {
2553                 (
2554                     \l__enumext_labelwidth_v_dim + \l__enumext_labelsep_v_dim
2555                 ) / \l__enumext_columns_v_int
2556                 - \l__enumext_listoffset_v_dim
2557             }
2558         }
2559         \dim_set_eq:NN \columnsep \l__enumext_columns_sep_v_dim

```

Then we will set the value of `\multicolsep` and `\columnseprule` equal to zero (we do not want a vertical rule in this environment).

```

2560         \skip_zero:N \multicolsep
2561         \dim_zero:N \columnseprule

```

We will calculate the *vertical spacing* settings for the `multicols` environment using the function `__enumext_keyans_multi_addvspace:` and apply our “*vertical adjust spacing*”, then start the `multicols` environment.

```

2562         \bool_if:NF \l__enumext_minipage_active_v_bool
2563         {
2564             \__enumext_keyans_multi_addvspace:
2565         }
2566         \raggedcolumns

```

```

2567         \begin{multicols}{\l__enumext_columns_v_int }
2568     }
2569 }

```

(End of definition for `__enumext_keyans_multicols_start:`)

`__enumext_keyans_multicols_stop:`

The function `__enumext_keyans_multicols_stop:` will stop the `multicols` environment. If the boolean variable `\l__enumext_minipage_active_v_bool` is false (not nested in `__enumext_mini-env*`) we will apply our vertical “adjust” spacing.

```

2570 \cs_new_protected:Nn \__enumext_keyans_multicols_stop:
2571 {
2572     \int_compare:nNtT { \l__enumext_columns_v_int } > { 1 }
2573     {
2574         \end{multicols}
2575         \bool_if:NF \l__enumext_minipage_active_v_bool
2576         {
2577             \par\addvspace{ \l__enumext_multicols_below_v_skip }
2578         }
2579     }
2580 }

```

(End of definition for `__enumext_keyans_multicols_stop:`)

`__enumext_after_list_v:`

The function `__enumext_after_list_v:` will check the state of the boolean variable `\l__enumext_minipage_active_v_bool`, if it is “true” a small test will be executed to check if we have omitted the use of `\miniright` (the `__enumext_mini-env*` environment has not been closed), then close `__enumext_mini-env*` and add the vertical adjustment space `\l__enumext_minipage_after_skip`, otherwise we will close the `multicols` environment.

```

2581 \cs_new_protected:Nn \__enumext_after_list_v:
2582 {
2583     \bool_if:NTF \l__enumext_minipage_active_v_bool
2584     {
2585         \int_compare:nNtT { \g__enumext_minipage_stat_int } = { 1 }
2586         {
2587             \msg_warning:nn { enumext } { missing-miniright }
2588             \miniright
2589         }
2590         \int_gzero:N \g__enumext_minipage_stat_int
2591         \end{\__enumext_mini-env*}
2592         \par\addvspace{ \l__enumext_minipage_after_skip }
2593     }
2594     { \__enumext_keyans_multicols_stop: }

```

Finally we will apply the `{\code}` handled by the `after` key together with the *vertical space* handled by the `below` key if they are present.

```

2595     \bool_set_false:N \l__enumext_keyans_env_bool
2596     \__enumext_after_stop_list_v:
2597     \__enumext_vspace_below_v:
2598 }

```

(End of definition for `__enumext_after_list_v:`)

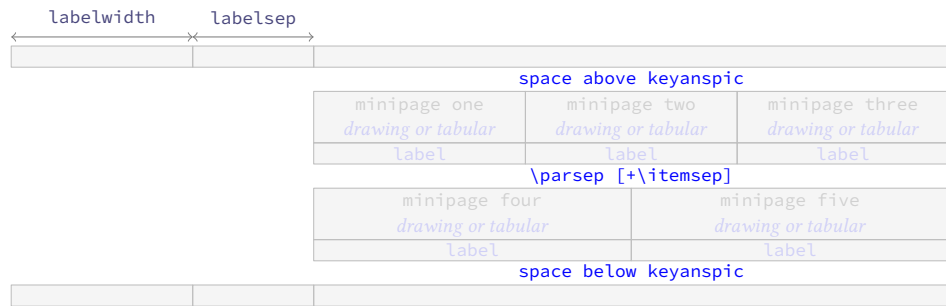
10.34 The environment `keyanspic` and `\anspic`

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

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

Again it is necessary to “adjust” the spacing, both vertical and horizontal, to obtain an output like the one shown in the figure 12.

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

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

10.34.1 The command `\anspic`

`\anspic` The `\anspic` command take three arguments, the starred (*) versions `\anspic*` and `\anspic*[\langle content \rangle]` store the current `\label` next to the `[\langle content \rangle]` if it is present in the `\langle sequence \rangle` and `\langle prop list \rangle` defined by `save-ans` key. This command is used as a replacement for `\item` in the `keyanspic` environment.

```
2599 \NewDocumentCommand \anspic { s o +m }
2600 {
```

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

```
2601   \bool_if:NF \l__enumext_store_active_bool
2602   {
2603     \msg_error:nnnn { enumext } { wrong-place } { keyanspic } { save-ans }
2604   }
2605   \int_compare:nNnT { \l__enumext_level_int } > { 1 }
2606   {
2607     \msg_error:nn { enumext } { keyanspic-wrong-level }
2608   }
2609   \int_compare:nNnT { \l__enumext_keyans_level_int } = { 1 }
2610   {
2611     \msg_error:nnnn { enumext } { command-wrong-place } { anspic } { keyans }
2612   }
```

The three arguments are handled by the function `__enumext_keyans_anspic_code:nnn` and stored in the sequence `\l__enumext_keyans_pic_body_seq` which is processed by the `keyanspic` environment.

```
2613   \seq_put_right:Nn \l__enumext_keyans_pic_body_seq
2614   {
2615     \__enumext_keyans_anspic_code:nnn { #1 } { #2 } { #3 }
2616   }
2617 }
```

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

`__enumext_keyans_anspic_code:nnn`

The function `__enumext_keyans_anspic_code:nnn` will be in charge of handling the “counter” and `\label`, which will have the same configuration as the `keyans` environment.

```
2618 \cs_new_protected:Nn \__enumext_keyans_anspic_code:nnn
2619 {
2620   \stepcounter { enumXvi }
2621   #3 \\\
2622   \bool_if:nT { #1 }
2623   {
2624     \__enumext_keyans_addto_prop:n { #2 }
2625     \__enumext_keyans_internal_ref:
2626     \__enumext_keyans_addto_seq:n { #2 }
2627     \bool_lazy_or:nnT
2628     { \l__enumext_show_answer_bool }
2629     { \l__enumext_show_position_bool }
2630     {
2631       \tl_set_eq:NN \l__enumext_label_v_tl \l__enumext_label_vi_tl
2632       \__enumext_keyans_show_left:n { #2 }
2633       \tl_set_eq:NN \l__enumext_label_vi_tl \l__enumext_label_v_tl
2634     }
2635   }
2636   \tl_use:N \l__enumext_label_font_style_v_tl
2637   \__enumext_wrapper_label_v:n { \__enumext_label_vi_tl }
2638 }
```

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

10.34.2 The environment keyanspic

`keyanspic` Now we define the environment `keyanspic` based on list. The optional argument [*number above, number below*] will determine the number of `minipage` environments that will be above and below separated by `\parsep+\itemsep` within it.

```
2639 \NewDocumentEnvironment{keyanspic}{ o }
2640 {
2641   \__enumext_keyans_pic_safe_exec:
2642   \__enumext_start_list:nn
2643   { }
2644   {
2645     \__enumext_keyans_pic_arg_two:
2646   }
```

We apply the “adjusted” vertical spacing above the environment

```
2647   \vspace { \__enumext_keyans_pic_above_skip }
2648 }
```

If the optional argument is not present, the number of times the `\anspic` command appears will be counted from `\l__enumext_keyans_pic_body_seq` and placed in `minipage` environments on a single line. Finally we check if `\anspic*` has been used, set the counter to zero and apply our “adjusted” vertical space below the environment.

```
2649 {
2650   \tl_if_novalue:nTF { #1 }
2651   {
2652     \__enumext_keyans_pic_do:e { \seq_count:N \l__enumext_keyans_pic_body_seq }
2653   }
2654   { \__enumext_keyans_pic_do:n { #1 } }
2655   \__enumext_stop_list:
2656   \__enumext_keyans_check_ans:nn { anspic } { keyanspic }
2657   \setcounter { enumXvi } { 0 }
2658   \vspace { \__enumext_topsep_v_skip }
2659   %\bool_set_false:N \l__enumext_store_active_bool
2660 }
```

(End of definition for `keyanspic`. This function is documented on page 12.)

`__enumext_keyans_pic_safe_exec:` The function `__enumext_keyans_pic_safe_exec:` check nested and level position inside the `enumext` environment.

```
2661 \cs_new_protected:Nn \__enumext_keyans_pic_safe_exec:
2662 {
2663   \int_incr:N \l__enumext_keyans_pic_level_int
2664   \int_compare:nNnT { \l__enumext_keyans_pic_level_int } > { 1 }
2665   {
2666     \msg_error:nn { enumext } { keyanspic-nested }
2667   }
2668 }
```

(End of definition for `__enumext_keyans_pic_safe_exec:`.)

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

```
2669 \cs_new_protected:Npn \__enumext_keyans_pic_skip_abs:N #1
2670 {
2671   \dim_compare:nNnT { #1 } < { 0pt }
2672   { \skip_set:Nn #1 { -#1 } }
2673 }
```

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

`__enumext_keyans_pic_arg_two:` The function `__enumext_keyans_pic_arg_two:` will be used in the second argument of the `__enumext_start_list:nn` function that defines the `keyanspic` environment, it will handle the setting of spaces.

```
2674 \cs_new_protected:Nn \__enumext_keyans_pic_arg_two:
2675 {
```

The first thing to do is to set the boolean variable `\l__enumext_leftmargin_tmp_v_bool` handled by the `list-indent` key to false, then we copy the definition of the second list argument from the `keyans` environment.

```
2676   \bool_set_false:N \l__enumext_leftmargin_tmp_v_bool
2677   \__enumext_list_arg_two_v:
```

We will add the value of `\itemsep` to `\parsep` which we will use as vertical spacing between the above and below `minipage` environments. and adjust the value of `\leftmargin`, the label and counter are handled directly by the `\anspic` command. Then we make equal to zero `\labelwidth`, `\labelsep`, `\partopsep` and `\itemsep` so that the horizontal and vertical spacing is not affected.

```

2678 \skip_add:Nn \parsep { \itemsep }
2679 \dim_add:Nn \leftmargin { -\labelwidth - \labelsep }
2680 \dim_zero:N \labelwidth
2681 \dim_zero:N \listparindent
2682 \dim_zero:N \labelsep
2683 \skip_zero:N \partopsep
2684 \skip_zero:N \itemsep

```

We set the value of `\l__enumext_keyans_pic_above_skip` which we will use to apply our “adjust” space above `keyanspic`, finally we call `__enumext_item_std:w` followed by `\scan_stop:` to prevent the error message returned by \TeX when not using the `\item` command.

```

2685 \__enumext_keyans_pic_skip_abs:N \parsep
2686 \skip_set:Nn \l__enumext_keyans_pic_above_skip
2687 {
2688   \box_dp:N \strutbox
2689   + \l__enumext_topsep_v_skip
2690   - \parsep
2691 }
2692 \__enumext_item_std:w \scan_stop:
2693 }

```

(End of definition for `__enumext_keyans_pic_arg_two:.`)

```

\__enumext_keyans_pic_do:n
\__enumext_keyans_pic_do:e

```

The optional argument is split by comma and is handled directly by the function `__enumext_keyans_pic_do:n` and passed to the function `__enumext_keyans_pic_row:n`.

```

2694 \cs_new_protected:Nn \__enumext_keyans_pic_do:n
2695 {
2696   \clist_map_function:nN { #1 } \__enumext_keyans_pic_row:n
2697 }
2698 \cs_generate_variant:Nn \__enumext_keyans_pic_do:n { e }

```

(End of definition for `__enumext_keyans_pic_do:n`.)

```
\__enumext_keyans_pic_row:n
```

The function `__enumext_keyans_pic_row:n` will set the widths for the `minipage` environments and place the content $\langle stored \rangle$ by `\anspic*` in the `\l__enumext_keyans_pic_body_seq` sequence inside them.

```

2699 \cs_new_protected:Nn \__enumext_keyans_pic_row:n
2700 {
2701   \dim_set:Nn \l__enumext_keyans_pic_width_dim { \linewidth / #1 }
2702   \int_set:Nn \l__enumext_keyans_pic_above_int { \l__enumext_keyans_pic_below_int }
2703   \int_set:Nn \l__enumext_keyans_pic_below_int { \l__enumext_keyans_pic_above_int + #1 }
2704   \int_step_inline:nnn
2705     { \l__enumext_keyans_pic_above_int + 1 }
2706     { \l__enumext_keyans_pic_below_int }
2707     {
2708       \__enumext_minipage:w [ b ]{ \l__enumext_keyans_pic_width_dim }
2709       \centering
2710       \seq_item:Nn \l__enumext_keyans_pic_body_seq { ##1 }
2711       \__enumext_endminipage:
2712     }
2713   \par
2714 }

```

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

10.35 The `enumext*` and `keyans*` environments

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

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

10.35.1 Functions for item box width

_enumext_starred_columns_set_vii:

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

```

2715 \cs_new_protected:Nn \__enumext_starred_columns_set_vii:
2716 {
2717   \dim_compare:nNnT { \l__enumext_columns_sep_vii_dim } = { \c_zero_dim }
2718   {
2719     \dim_set:Nn \l__enumext_columns_sep_vii_dim
2720     {
2721       ( \l__enumext_labelwidth_vii_dim + \l__enumext_labelsep_vii_dim )
2722       / \l__enumext_columns_vii_int
2723     }
2724   }
2725   \int_set:Nn \l__enumext_tmpa_vii_int { \l__enumext_columns_vii_int - \c_one_int }
2726   \dim_set:Nn \l__enumext_item_width_vii_dim
2727   {
2728     ( \linewidth - \l__enumext_columns_sep_vii_dim * \l__enumext_tmpa_vii_int )
2729     / \l__enumext_columns_vii_int - \l__enumext_labelwidth_vii_dim
2730     - \l__enumext_labelsep_vii_dim
2731   }
2732   \dim_zero_new:N \itemwidth
2733 }

```

(End of definition for _enumext_starred_columns_set_vii:.)

_enumext_starred_joined_item_vii:n

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

```

2734 \cs_new_protected:Npn \__enumext_starred_joined_item_vii:n #1
2735 {
2736   \int_set:Nn \l__enumext_joined_item_vii_int {#1}
2737   \int_compare:nNnT { \l__enumext_joined_item_vii_int } > { \l__enumext_columns_vii_int }
2738   {
2739     \msg_warning:nnee { enumext } { item-joined }
2740     { \int_use:N \l__enumext_joined_item_vii_int }
2741     { \int_use:N \l__enumext_columns_vii_int }
2742     \int_set:Nn \l__enumext_joined_item_vii_int
2743     {
2744       \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + \c_one_int
2745     }
2746   }
2747   \int_compare:nNnT
2748   { \l__enumext_joined_item_vii_int }
2749   >
2750   { \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + \c_one_int }
2751   {
2752     \msg_warning:nnee { enumext } { item-joined-columns }
2753     { \int_use:N \l__enumext_joined_item_vii_int }
2754     {
2755       \int_eval:n
2756       { \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + \c_one_int }
2757     }
2758     \int_set:Nn \l__enumext_joined_item_vii_int
2759     {
2760       \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + \c_one_int
2761     }
2762   }

```

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

```

2763   \int_compare:nNnTF { \l__enumext_joined_item_vii_int } > { \c_one_int }
2764   {
2765     \int_set_eq:NN \l__enumext_joined_item_aux_vii_int \l__enumext_joined_item_vii_int
2766     \int_decr:N \l__enumext_joined_item_aux_vii_int
2767     \int_add:Nn \l__enumext_item_column_pos_vii_int { \l__enumext_joined_item_aux_vii_int }
2768     \int_gadd:Nn \g__enumext_item_count_all_vii_int { \l__enumext_joined_item_aux_vii_int }
2769     \dim_set:Nn \l__enumext_joined_width_vii_dim
2770     {
2771       \l__enumext_item_width_vii_dim * \l__enumext_joined_item_vii_int
2772       + ( \l__enumext_labelwidth_vii_dim + \l__enumext_labelsep_vii_dim
2773         + \l__enumext_columns_sep_vii_dim
2774         ) * \l__enumext_joined_item_aux_vii_int

```

```

2775     }
2776     \dim_set_eq:NN \itemwidth \l__enumext_joined_width_vii_dim
2777   }
2778   {
2779     \dim_set_eq:NN \l__enumext_joined_width_vii_dim \l__enumext_item_width_vii_dim
2780     \dim_set_eq:NN \itemwidth \l__enumext_item_width_vii_dim
2781   }
2782 }

```

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

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

```

2783 \cs_new_protected:Nn \__enumext_start_mini_vii:
2784 {
2785   \dim_compare:nNnT { \l__enumext_minipage_right_vii_dim } > { \c_zero_dim }
2786   {
2787     \dim_set:Nn \l__enumext_minipage_left_vii_dim
2788     {
2789       \linewidth
2790       - \l__enumext_minipage_right_vii_dim
2791       - \l__enumext_minipage_hsep_vii_dim
2792     }
2793     \bool_set_true:N \l__enumext_minipage_active_vii_bool
2794     \dim_gset_eq:NN
2795       \g__enumext_minipage_right_vii_dim
2796       \l__enumext_minipage_right_vii_dim
2797     \__enumext_mini_addvspace_vii:
2798     \nointerlineskip\noindent
2799     \begin{__enumext_mini_env*}{ \l__enumext_minipage_left_vii_dim }
2800   }
2801 }

```

(End of definition for `__enumext_start_mini_vii:`.)

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

```

2802 \cs_new_protected:Nn \__enumext_stop_mini_vii:
2803 {
2804   \bool_if:NT \l__enumext_minipage_active_vii_bool
2805   {
2806     \end{__enumext_mini_env*}
2807     \hfill
2808     \bool_gset_true:N \g__enumext_minipage_active_vii_bool
2809   }
2810 }

```

Finally we execute code passed to the `miniright` key stored in the variable `\g__enumext_miniright_code_vii_tl` in the `__enumext_mini_env*` environment on the “*right side*”.

```

2811 \__enumext_after_env:nn {enumext*}
2812 {
2813   \bool_if:NT \g__enumext_minipage_active_vii_bool
2814   {
2815     \begin{__enumext_mini_env*}{ \g__enumext_minipage_right_vii_dim }
2816     \par\addvspace { \g__enumext_minipage_right_skip }
2817     \bool_if:NF \g__enumext_minipage_center_vii_bool
2818     {
2819       \centering
2820     }
2821     \tl_use:N \g__enumext_miniright_code_vii_tl % the code
2822     \end{__enumext_mini_env*}
2823     \par\addvspace{ \g__enumext_minipage_after_skip }
2824   }
2825   \bool_gset_false:N \g__enumext_minipage_active_vii_bool
2826   \bool_gset_true:N \g__enumext_minipage_center_vii_bool
2827   \tl_gclear:N \g__enumext_miniright_code_vii_tl
2828   \dim_gzero:N \g__enumext_minipage_right_vii_dim
2829 }

```


(End of definition for `__enumext_stop_mini_vii:`.)

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

```

2830 \NewDocumentEnvironment{enumext*}{o }
2831 {
2832   \__enumext_safe_exec_vii:
2833   \__enumext_parse_keys_vii:n {#1}
2834   \__enumext_before_list_vii:
2835   \__enumext_start_store_level_vii:
2836   \__enumext_start_list:nn { }
2837   {
2838     \__enumext_list_arg_two_vii:
2839     \__enumext_before_keys_exec_vii:
2840   }
2841   \__enumext_starred_columns_set_vii:
2842   \item[] \scan_stop:
2843   \cs_set_eq:NN \__enumext_stop_item_tmp_vii: \noindent
2844   \cs_set_eq:NN \item \__enumext_start_item_tmp_vii:
2845 }
2846 {
2847   \__enumext_stop_item_tmp_vii:
2848   \__enumext_remove_extra_parsep_vii:
2849   \__enumext_stop_list:
2850   \__enumext_stop_store_level_vii:
2851   \__enumext_after_list_vii:
2852 }
```

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

`__enumext_safe_exec_vii:` First check the maximum nesting level for the `enumext*` environment then set the vars `\l__enumext_starred_bool` and `\g__enumext_starred_bool`.

```

2853 \cs_new_protected:Nn \__enumext_safe_exec_vii:
2854 {
2855   \int_incr:N \l__enumext_level_h_int
2856   \int_compare:nNnT { \l__enumext_level_h_int } > { 1 }
2857   {
2858     \msg_error:nn { enumext } { nested }
2859   }
2860   \bool_set_true:N \l__enumext_starred_bool
2861   \bool_lazy_all:nT
2862   {
2863     { \bool_not_p:n { \l__enumext_standar_bool } }
2864     { \int_compare_p:nNn { \l__enumext_level_int } = { \c_zero_int } }
2865   }
2866   {
2867     \bool_gset_true:N \g__enumext_starred_bool
2868   }
2869 }
```

(End of definition for `__enumext_safe_exec_vii:`.)

`__enumext_parse_keys_vii:n` Parse [`<key = val>`] for `enumext*`. If the variable `\l__enumext_store_active_bool` is true it will call the function `__enumext_parse_store_keys_vii:n` and reprocess the keys to pass them to the storage sequence.

```

2870 \cs_new_protected:Npn \__enumext_parse_keys_vii:n #1
2871 {
2872   \tl_if_novalue:nF {#1}
2873   {
2874     \keys_set:nn { enumext / enumext* } {#1}
2875     \bool_if:NT \l__enumext_store_active_bool
2876     {
2877       \__enumext_parse_store_keys_vii:n {#1}
2878     }
2879   }
2880 }
```

(End of definition for `__enumext_parse_keys_vii:n`.)

__enumext_parse_store_keys_vii:n

The function __enumext_parse_store_keys_vii:n searches for the values of the `columns` and `columns-sep` keys in the optional argument in `enumext*` environment as long as the starred versions of the `columns*` and `columns-sep*` keys are not active. The captured values are stored in the variable \l__enumext_store_opt_vii_tl which is used by the function __enumext_store_level_open_vii:.

```

2881 \cs_new_protected:Npn \__enumext_parse_store_keys_vii:n #1
2882 {
2883   \bool_if:NF \l__enumext_store_columns_vii_bool
2884   {
2885     \regex_match:nnT { \b columns\b } {#1}
2886     {
2887       \int_set_eq:NN
2888         \l__enumext_store_columns_vii_int
2889         \l__enumext_columns_vii_int
2890       \tl_put_right:Ne \l__enumext_store_opt_vii_tl
2891         {
2892           columns = \exp_not:V \l__enumext_store_columns_vii_int ,
2893         }
2894     }
2895   }
2896   \bool_if:NF \l__enumext_store_columns_sep_vii_bool
2897   {
2898     \regex_match:nnT { \b columns-sep\b } {#1}
2899     {
2900       \dim_set_eq:NN
2901         \l__enumext_store_columns_sep_vii_dim
2902         \l__enumext_columns_sep_vii_dim
2903       \tl_put_right:Ne \l__enumext_store_opt_vii_tl
2904         {
2905           columns-sep = \exp_not:V \l__enumext_store_columns_sep_vii_dim,
2906         }
2907     }
2908   }
2909 }

```

(End of definition for __enumext_parse_store_keys_vii:n.)

__enumext_before_list_vii:

The function __enumext_before_list_vii: will add the vertical spacing on the environment if the `above` key is active next to the `{\code}` defined by the `before*` key if it is active, the call the function __enumext_start_mini_vii: handle by `mini-env`.

```

2910 \cs_new_protected:Nn \__enumext_before_list_vii:
2911 {
2912   \__enumext_vspace_above_vii:
2913   \__enumext_before_args_exec_vii:
2914   \__enumext_start_mini_vii:
2915 }

```

(End of definition for __enumext_before_list_vii:.)

__enumext_after_list_vii:

The function __enumext_after_list: first call the function __enumext_stop_mini_vii:, then apply the `{\code}` handled by the `after` key together with the *vertical space* handled by the `below` key if they are present. Finally set false the vars \g__enumext_starred_bool and \l__enumext_starred_bool, save the *current value* of the counter in \g__enumext_resume_vii_int for the `resume` key. If the `save-ans` key is active, it will create the integer variable for the `resume` key, we only have to assign it the value of the current counter.

```

2916 \cs_new_protected:Nn \__enumext_after_list_vii:
2917 {
2918
2919   \__enumext_stop_mini_vii:
2920   \__enumext_after_stop_list_vii:
2921   \__enumext_vspace_below_vii:
2922   \int_gset_eq:NN \g__enumext_resume_vii_int \value{enumXvii}
2923   \int_if_exist:cT { g__enumext_resume_ \l__enumext_store_name_tl _int }
2924   {
2925     \int_gset_eq:cN
2926       { g__enumext_resume_ \l__enumext_store_name_tl _int }
2927     { \value{enumXvii} }
2928   }
2929   \bool_lazy_and:nnT { \g__enumext_starred_bool } { \l__enumext_check_ans_bool }
2930   {

```

```

2931         \bool_gset_true:N \g__enumext_check_ans_show_h_bool
2932         \tl_gset:NV \g__enumext_store_name_tl \l__enumext_store_name_tl
2933     }
2934     \bool_gset_false:N \g__enumext_starred_bool
2935     \bool_set_false:N \l__enumext_starred_bool
2936 }

```

(End of definition for `__enumext_after_list_vii:`)

`__enumext_start_store_level_vii:`
`__enumext_stop_store_level_vii:`

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

```

2937 \cs_new_protected:Nn \__enumext_start_store_level_vii:
2938 {
2939     \bool_if:NT \l__enumext_store_active_bool
2940     {
2941         \int_compare:nNnT { \l__enumext_level_int } > { \c_zero_int }
2942         {
2943             \__enumext_store_level_open_vii:
2944         }
2945     }
2946 }
2947 \cs_new_protected:Nn \__enumext_stop_store_level_vii:
2948 {
2949     \bool_if:NT \l__enumext_store_active_bool
2950     {
2951         \int_compare:nNnT { \l__enumext_level_int } > { \c_zero_int }
2952         {
2953             \__enumext_store_level_close_vii:
2954         }
2955     }
2956 }

```

(End of definition for `__enumext_start_store_level_vii:` and `__enumext_stop_store_level_vii:`)

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

`__enumext_start_item_tmp_vii:`

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

```

2957 \cs_new_protected_nopar:Nn \__enumext_start_item_tmp_vii:
2958 {
2959     \__enumext_stop_item_tmp_vii:
2960     \int_incr:N \l__enumext_item_column_pos_vii_int
2961     \int_gincr:N \g__enumext_item_count_all_vii_int
2962     \__enumext_item_peek_args_vii:
2963 }

```

(End of definition for `__enumext_start_item_tmp_vii:`)

`__enumext_item_peek_args_vii:`

The function `__enumext_item_peek_args_vii:` will handle the `\item(number)`. Look for the argument “(”, if it is present we will call the function `__enumext_joined_item_vii:w(number)`, which is in charge of joining the item's in the same row, in case they are not present we will set the default value (1).

```

2964 \cs_new_protected:Nn \__enumext_item_peek_args_vii:
2965 {
2966     \peek_meaning:NTF (
2967         { \__enumext_joined_item_vii:w }
2968         { \__enumext_joined_item_vii:w (1) }
2969     }

```

(End of definition for `__enumext_item_peek_args_vii:`)

`__enumext_joined_item_vii:w`

The function `__enumext_joined_item_vii:w` will first call the function `__enumext_starred_joined_item_vii:n` in charge of setting the *width* of the box that will store the content passed to `\item`. Then we will look for the argument “*”, if it is present we will call the function `__enumext_starred_item_vii:w` otherwise we will call the function `__enumext_standard_item_vii:w`.

```

2970 \cs_new_protected:Npn \__enumext_joined_item_vii:w (#1)
2971 {

```

```

2972     \__enumext_starred_joined_item_vii:n {#1}
2973     \peek_meaning_remove:NTF *
2974     { \__enumext_starred_item_vii:w }
2975     { \__enumext_standard_item_vii:w }
2976 }

```

(End of definition for __enumext_joined_item_vii:w.)

__enumext_standard_item_vii:w

The function __enumext_standard_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].

```

2977 \cs_new_protected:Npn \__enumext_standard_item_vii:w
2978 {
2979     \bool_set_false:N \l__enumext_item_starred_vii_bool
2980     \peek_meaning:NTF [
2981     {
2982         \bool_set_eq:NN
2983         \l__enumext_wrap_label_vii_bool
2984         \l__enumext_wrap_label_opt_vii_bool
2985         \__enumext_start_item_vii:w
2986     }
2987     {
2988         \bool_set_true:N \l__enumext_wrap_label_vii_bool
2989         \legacy_if_set_true:n { @noitemarg }
2990         \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
2991     }
2992 }

```

(End of definition for __enumext_standard_item_vii:w.)

__enumext_starred_item_vii:w

The function __enumext_starred_item_vii:w together with the specified auxiliary functions `aux_i:w`, `aux_ii:w`, and `aux_iii:w` execute \item*, \item*[⟨*symbol*⟩] and \item*[⟨*symbol*⟩][⟨*offset*⟩].

__enumext_starred_item_vii_aux_i:w

__enumext_starred_item_vii_aux_ii:w

__enumext_starred_item_vii_aux_iii:w

```

2993 \cs_new_protected:Npn \__enumext_starred_item_vii:w
2994 {
2995     \bool_set_true:N \l__enumext_item_starred_vii_bool
2996     \bool_set_true:N \l__enumext_wrap_label_vii_bool
2997     \peek_meaning:NTF [
2998     { \__enumext_starred_item_vii_aux_i:w }
2999     { \__enumext_starred_item_vii_aux_ii:w }
3000 }
3001 \cs_new_protected:Npn \__enumext_starred_item_vii_aux_i:w [#1]
3002 {
3003     \tl_gset:Nn \g__enumext_item_symbol_aux_vii_tl {#1}
3004     \__enumext_starred_item_vii_aux_ii:w
3005 }
3006 \cs_new_protected:Npn \__enumext_starred_item_vii_aux_ii:w
3007 {
3008     \peek_meaning:NTF [
3009     { \__enumext_starred_item_vii_aux_iii:w }
3010     {
3011         \dim_set_eq:NN
3012         \l__enumext_item_symbol_sep_vii_dim
3013         \l__enumext_labelsep_vii_dim
3014         \legacy_if_set_true:n { @noitemarg }
3015         \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
3016     }
3017 }
3018 \cs_new_protected:Npn \__enumext_starred_item_vii_aux_iii:w [#1]
3019 {
3020     \dim_set:Nn \l__enumext_item_symbol_sep_vii_dim {#1}
3021     \legacy_if_set_true:n { @noitemarg }
3022     \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
3023 }

```

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

Real definition of `\item`

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

`__enumext_start_item_vii:w`

The first thing we will do is set the value of `__enumext_stop_item_tmp_vii:` equal to the value of `__enumext_stop_item_vii:` which we will define later and add the `hyperref` compatible `enumXvii` counter, after that we will start capturing the item content in a box. Here need setting the `\if@hyper@item` switch to “true” for `hyperref` compatible. The explanation for this is given by the master Heiko Oberdiek on `\refstepcounter{enumi}` twice (or more) creates destination with the same identifier.

```

3024 \cs_new_protected_nopar:Npn \__enumext_start_item_vii:w [#1]
3025 {
3026   \cs_set_eq:NN \__enumext_stop_item_tmp_vii: \__enumext_stop_item_vii:
3027   \legacy_if:nT { @noitemarg }
3028   {
3029     \legacy_if_set_false:n { @noitemarg }
3030     \legacy_if:nT { @nmbrrlist }
3031     {
3032       \bool_if:NT \l__enumext_hyperref_bool
3033       {
3034         \legacy_if_set_true:n { @hyper@item }
3035       }
3036       \refstepcounter{enumXvii}
3037       % code for check-ans
3038       \bool_if:NT \l__enumext_check_ans_bool
3039       {
3040         % If true |no-store| key => nested in |enumext|
3041         \bool_if:NTF \l__enumext_store_ans_bool
3042         {
3043           \int_gadd:cn { g__enumext_count_item_ \__enumext_level: _int }
3044           { \int_use:c { g__enumext_count_level_ \__enumext_level: _int } + 1 }
3045         }
3046         {
3047           \int_gincr:N \g__enumext_count_item_all_int
3048           \int_gincr:N \g__enumext_count_level_vii_int
3049         }
3050       }
3051     }
3052   }

```

Here we start capturing `\item` and its contents into a group using the plain form of the `lrbox` environment. If the state of the variable `\l__enumext_footnotes_key_bool` is false, we will redefine the command `\footnote`, followed by printing the $\langle symbol \rangle$ defined for `\item*` if it is present and open a new group inside which we execute `font` key next to `\item` and the keys `wrap-label`, `wrap-label*`, `align`, close the group and execute the key `labelsep` and then the key `first`. Finally we open the `minipage` environment and execute the `listparindent` key which will be equal to `\parindent`, the `parsep` key which will be equal to `\parskip` and the `itemindent` key.

```

3053 \group_begin:
3054 \lrbox{ \l__enumext_item_text_vii_box }
3055 \bool_if:NF \l__enumext_footnotes_key_bool
3056 {
3057   \__enumext_renew_footnote:
3058 }
3059 \bool_if:NT \l__enumext_item_starred_vii_bool
3060 {
3061   \tl_if_blank:VT \g__enumext_item_symbol_aux_vii_tl
3062   {
3063     \tl_gset_eq:NN
3064     \g__enumext_item_symbol_aux_vii_tl \l__enumext_item_symbol_vii_tl
3065   }
3066   \mode_leave_vertical:
3067   \skip_horizontal:n { -\l__enumext_item_symbol_sep_vii_dim }
3068   \makebox[ \opt ][ r ]{ \g__enumext_item_symbol_aux_vii_tl }
3069   \skip_horizontal:N \l__enumext_item_symbol_sep_vii_dim
3070   \tl_gclear:N \g__enumext_item_symbol_aux_vii_tl
3071 }
3072 \group_begin:
3073 \tl_use:N \l__enumext_label_font_style_vii_tl
3074 \bool_if:NTF \l__enumext_wrap_label_vii_bool
3075 {
3076   \makebox[ \l__enumext_labelwidth_vii_dim ][ \l__enumext_align_label_vii_str ]

```

```

3077         { \__enumext_wrapper_label_vii:n {#1} }
3078     }
3079     {
3080         \makebox[ \__enumext_labelwidth_vii_dim ][ \__enumext_align_label_vii_str ]{ #1 }
3081     }
3082     \group_end:
3083     \skip_horizontal:N \__enumext_labelsep_vii_dim
3084     \tl_use:N \__enumext_after_list_args_vii_tl
3085     \__enumext_minipage:w [ t ]{ \__enumext_joined_width_vii_dim }
3086         \skip_set_eq:NN \parindent \__enumext_listparindent_vii_dim
3087         \skip_set_eq:NN \parskip \__enumext_parsep_vii_skip
3088         \tl_use:N \__enumext_fake_item_indent_vii_tl
3089     }

```

(End of definition for __enumext_start_item_vii:w.)

__enumext_stop_item_vii: The function __enumext_stop_item_vii: shall terminate with the capture of \item and its *contents*. Close the environments minipage, lrbox and the group. Then we only have to set the width of the box and print it next to \footnote, and add the horizontal and vertical separation between the boxes.

```

3090 \cs_new_protected_nopar:Nn \__enumext_stop_item_vii:
3091 {
3092     \__enumext_endminipage:
3093     \endlrbox
3094     \group_end:
3095     \box_set_wd:Nn \__enumext_item_text_vii_box
3096     {
3097         \__enumext_joined_width_vii_dim
3098         + \__enumext_labelwidth_vii_dim
3099         + \__enumext_labelsep_vii_dim
3100     }
3101     \int_set:Nn \hbadness { 10000 }
3102     \box_use:N \__enumext_item_text_vii_box
3103     \bool_if:NF \__enumext_footnotes_key_bool
3104     {
3105         \__enumext_print_footnote:
3106     }
3107     \int_compare:nNnTF { \__enumext_item_column_pos_vii_int } = { \__enumext_columns_vii_int }
3108     {
3109         \par\noindent
3110         \int_zero:N \__enumext_item_column_pos_vii_int
3111     }
3112     { \hspace{ \__enumext_columns_sep_vii_dim } }
3113 }

```

(End of definition for __enumext_stop_item_vii:.)

__enumext_remove_extra_parsep_vii: Finally we will remove the vertical space equal to \parsep when the total number of items is divisible by the number of items in the last row of the environment.

```

3114 \cs_new_protected:Nn \__enumext_remove_extra_parsep_vii:
3115 {
3116     \int_compare:nNnT
3117     {
3118         \int_mod:nn { \g__enumext_item_count_all_vii_int } { \__enumext_columns_vii_int }
3119     }
3120     =
3121     { \c_zero_int }
3122     {
3123         \par
3124         \vspace{ -\__enumext_itemsep_vii_skip }
3125         \int_gzero:N \g__enumext_item_count_all_vii_int
3126     }
3127 }

```

(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.

```

3128 \__enumext_after_env:nn {enumext*}
3129 {
3130     \bool_if:NT \g__enumext_check_ans_show_h_bool
3131     {

```

```

3132         \int_compare:nNnT { \l__enumext_level_int } = { 0 }
3133         {
3134             \__enumext_check_ans_active_vii:
3135         }
3136     }
3137     \bool_gset_false:N \g__enumext_check_ans_show_h_bool
3138     \tl_gclear:N \g__enumext_store_name_tl
3139 }

```

10.36 The command \getkeyans

`\getkeyans` The `\getkeyans` command takes a mandatory argument of the form $\langle store\ name : position \rangle$. Retrieve a “single” content stored by `\anskey`, `\anspic*` and `\item*` from $\langle prop\ list \rangle$ defined by `save-ans` key.

```

3140 \NewDocumentCommand \getkeyans { m }
3141 {
3142     \exp_args:Ne \__enumext_getkeyans_aux:n
3143     { \tl_to_str:e { \text_expand:n {#1} } }
3144 }

```

(End of definition for `\getkeyans`. This function is documented on page 13.)

`__enumext_getkeyans_aux:n` The internal function `__enumext_getkeyans_aux:n` is in charge of *splitting* the $\langle argument \rangle$ using “:”. If “:” is omitted it will return an error.

```

3145 \cs_new_protected:Npn \__enumext_getkeyans_aux:n #1
3146 {
3147     \str_if_in:nnTF {#1} { : }
3148     {
3149         \use:e
3150         {
3151             \cs_set:Npn \exp_not:N \__enumext_tmp:w #1 \c_colon_str ##2 \scan_stop:
3152             { {##1} {##2} }
3153         }
3154         \exp_after:wN \__enumext_getkeyans:nn \__enumext_tmp:w #1 \scan_stop:
3155     }
3156     { \msg_error:nnn { enumext } { missing-colon } {#1} }
3157 }

```

(End of definition for `__enumext_getkeyans_aux:n`.)

`__enumext_getkeyans:nn` The internal function `__enumext_getkeyans:nn` will check for the existence of the $\langle prop\ list \rangle$, if it does not exist it will return an error message, then it will fetch the content specified by the second $\langle argument \rangle$ from $\langle prop\ list \rangle$.

```

3158 \cs_new_protected:Npn \__enumext_getkeyans:nn #1 #2
3159 {
3160     \prop_if_exist:cF { g__enumext_#1_prop }
3161     { \msg_error:nnn { enumext } { undefined-storage-anskey } {#1} }
3162     \group_begin:
3163         \prop_item:cn { g__enumext_#1_prop }{#2}
3164     \group_end:
3165 }

```

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

10.37 The command \printkeyans

The `\printkeyans` command prints “all stored content” in the $\langle sequence \rangle$ defined by the `save-ans` key. The first thing we will do is to define a set of $\langle keys \rangle$ with which we will control the options of the different nesting levels for the `enumext` and `enumext*` environment by storing the values of these in the token list variables `\l__enumext_print_keyans_X_tl`.

```

3166 \keys_define:nn { keyanskey / print }
3167 {
3168     level-1 .code:n      = \tl_put_right:Nn \l__enumext_print_keyans_i_tl
3169                         {
3170                             \setenumext[level,1] {#1} \setenumext[print,1] {#1}
3171                         },
3172     level-1 .initial:n   = { label=\arabic*. , nosep, columns=2, first=\small, font=\small },
3173     level-2 .code:n      = \tl_put_right:Nn \l__enumext_print_keyans_ii_tl
3174                         {
3175                             \setenumext[level,2] {#1} \setenumext[print,2] {#1}
3176                         },
3177     level-2 .initial:n   = { nosep, label=(\alph*), first=\small, font=\small },

```



```

3178   level-3 .code:n      = \tl_put_right:Nn \l__enumext_print_keyans_iii_tl
3179                       {
3180                         \setenumext[level,3] {#1} \setenumext[print,3] {#1}
3181                       },
3182   level-3 .initial:n   = { nosep, label=\roman*., first=\small, font=\small },
3183   level-4 .code:n      = \tl_put_right:Nn \l__enumext_print_keyans_iv_tl
3184                       {
3185                         \setenumext[level,4] {#1} \setenumext[print,4] {#1}
3186                       },
3187   level-4 .initial:n   = { nosep, label=\Alph*., first=\small, font=\small },
3188   level-* .code:n      = \tl_put_right:Nn \l__enumext_print_keyans_vii_tl % starred
3189                       {
3190                         \setenumext[enumext*] {#1} %\setenumext[print,*] {#1}
3191                       },
3192   level-* .initial:n   = { label=\arabic*., nosep, columns=2, first=\small, font=\small },
3193   }

```

`\printkeyans` Create a user command to print “all stored content” in *⟨sequence⟩* for `\anskey`, `\item*` and `\anspic*`.

```

3194 \NewDocumentCommand \printkeyans { s O{} m }
3195 {
3196   \group_begin:
3197     \tl_use:N \l__enumext_print_keyans_i_tl
3198     \tl_use:N \l__enumext_print_keyans_ii_tl
3199     \tl_use:N \l__enumext_print_keyans_iii_tl
3200     \tl_use:N \l__enumext_print_keyans_iv_tl
3201     \tl_use:N \l__enumext_print_keyans_vii_tl
3202     \__enumext_printkeyans:nnn { #1 } { #2 } { #3 }
3203   \group_end:
3204 }

```

(End of definition for `\printkeyans`. This function is documented on page 13.)

`__enumext_printkeyans:nnn` The internal function `__enumext_printkeyans:nnn` will check for the existence of the *⟨sequence⟩*, if it does not exist it will return an error message, then it will fetch the content specified by the first argument mapping the *⟨sequence⟩*.

#1 : starred

#2 : key-val

#3 : seq-name

```

3205 \cs_new_protected:Npn \__enumext_printkeyans:nnn #1 #2 #3
3206 {
3207   \seq_if_exist:cTF { g__enumext_#3_seq }
3208   {
3209     \seq_if_empty:cF { g__enumext_#3_seq }
3210     {
3211       \bool_if:nTF {#1}
3212       {
3213         \begin{enumext*}[#2]
3214         \seq_map_inline:cn { g__enumext_#3_seq } { ##1 }
3215         \end{enumext*}
3216       }
3217       {
3218         \begin{enumext}[#2]
3219         \seq_map_inline:cn { g__enumext_#3_seq } { ##1 }
3220         \end{enumext}
3221       }
3222     }
3223   }
3224   {
3225     \msg_error:nnn { enumext } { undefined-storage-anskey } {#3}
3226   }
3227 }

```

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

10.38 The command `\setenumext`

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

```

3228 \keys_define:nn { enumext / meta-families }
3229 {
3230   level-1 .code:n = { \keys_set:nn { enumext / level-1 } {#1} },
3231   level-2 .code:n = { \keys_set:nn { enumext / level-2 } {#1} },

```

```

3232   level-3   .code:n = { \keys_set:nn { enumext / level-3 } {#1} } ,
3233   level-4   .code:n = { \keys_set:nn { enumext / level-4 } {#1} } ,
3234   keyans    .code:n = { \keys_set:nn { enumext / keyans   } {#1} } ,
3235   enumext*   .code:n = { \keys_set:nn { enumext / enumext* } {#1} } ,
3236   keyans*    .code:n = { \keys_set:nn { enumext / keyans*  } {#1} } ,
3237   print-1   .code:n = { \keys_set:nn { keyanskey / print } { level-1 = {#1} } } ,
3238   print-2   .code:n = { \keys_set:nn { keyanskey / print } { level-2 = {#1} } } ,
3239   print-3   .code:n = { \keys_set:nn { keyanskey / print } { level-3 = {#1} } } ,
3240   print-4   .code:n = { \keys_set:nn { keyanskey / print } { level-4 = {#1} } } ,
3241   print-*   .code:n = { \keys_set:nn { keyanskey / print } { level-* = {#1} } } ,
3242   unknown   .code:n = { \msg_error:nn { enumext } { unknown-key-family } } ,
3243 }

```

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

```

3244 \seq_const_from_clist:Nn \c__enumext_all_families_seq
3245 {
3246   level-1 , level-2 , level-3 , level-4 , keyans , enumext* ,
3247   keyans* , print-1 , print-2 , print-3 , print-4 , print-* ,
3248 }

```

`\setenumext` Now we define the user command `\setenumext`.

```

3249 \NewDocumentCommand \setenumext { o +m }
3250 {
3251   \tl_if_novalue:nTF {#1}
3252   {
3253     \seq_map_inline:Nn \c__enumext_all_families_seq
3254   }
3255   {
3256     \seq_clear:N \l__enumext_setkey_tmpa_seq
3257     \seq_set_from_clist:Nn \l__enumext_setkey_tmpb_seq {#1}
3258     \int_set:Nn \l__enumext_setkey_tmpa_int
3259     {
3260       \seq_count:N \l__enumext_setkey_tmpb_seq
3261     }
3262     \int_compare:nNnTF { \l__enumext_setkey_tmpa_int } > { 1 }
3263     {
3264       \seq_pop_left:NN \l__enumext_setkey_tmpb_seq \l__enumext_setkey_tmpa_tl
3265       \seq_map_function:NN \l__enumext_setkey_tmpb_seq \l__enumext_set_parse:n
3266       \seq_set_map_e:NNn \l__enumext_setkey_tmpa_seq \l__enumext_setkey_tmpa_seq
3267       {
3268         \tl_use:N \l__enumext_setkey_tmpa_tl - ##1
3269       }
3270     }
3271     {
3272       \seq_put_right:Ne \l__enumext_setkey_tmpa_seq { \tl_trim_spaces:n {#1} }
3273     }
3274     \seq_if_empty:NTF \l__enumext_setkey_tmpa_seq
3275     { \seq_map_inline:Nn \c__enumext_all_families_seq }
3276     { \seq_map_inline:Nn \l__enumext_setkey_tmpa_seq }
3277   }
3278   {
3279     \keys_set:nn { enumext / meta-families } { ##1 = {#2} }
3280   }
3281 }

```

(End of definition for `\setenumext`. This function is documented on page 5.)

`__enumext_set_parse:n`
`__enumext_set_error:nn`

Internal functions used by the `\setenumext` command.

```

3282 \cs_new_protected:Npn \__enumext_set_parse:n #1
3283 {
3284   \tl_set:Ne \l__enumext_setkey_tmpb_tl { \tl_trim_spaces:n {#1} }
3285   \int_step_inline:nnn { 0 } { 4 } %<- max level
3286   { \tl_remove_all:Nn \l__enumext_setkey_tmpb_tl {##1} }
3287   \tl_if_empty:NTF \l__enumext_setkey_tmpb_tl
3288   {
3289     \seq_put_right:Ne \l__enumext_setkey_tmpa_seq
3290     { \tl_trim_spaces:n {#1} }
3291   }
3292   { \__enumext_set_error:nn {#1} { } }
3293 }
3294 \cs_new_protected:Npn \__enumext_set_error:nn #1#2
3295 { \msg_error:nnn { enumext } { invalid-key } {#1} {#2} }

```

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

10.39 Messages

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

```

3296 \msg_new:nnn { enumext } { package-load }
3297 {
3298   The ~ '#1' ~ package ~ is ~ already ~ loaded.
3299 }

3300 \msg_new:nnn { enumext } { package-not-load }
3301 {
3302   The ~ '#1' ~ package ~ will ~ be ~ loaded ~ as ~ a ~ dependency.
3303 }

3304 \msg_new:nnn { enumext } { package-load-foot }
3305 {
3306   The ~ '#1' ~ package ~ is ~ loaded ~ with ~ the ~ option ~ '#2'.
3307 }
```

Message used in the creation of counters by `enumext` package.

```

3308 \msg_new:nnn { enumext } { counters }
3309 {
3310   The ~ counter ~ '#1' ~ is ~ already ~ defined ~ by ~ some ~ \\
3311   package ~ or ~ macro, ~ it ~ cannot ~ be ~ continued.
3312 }
```

Message used by `[<key = val>]` system and `\setenumext` command.

```

3313 \msg_new:nnn { enumext } { invalid-key }
3314 {
3315   The ~ key ~ '#1' ~ is ~ not ~ know ~ the ~ level ~ #2.
3316 }
3317 \msg_new:nnn { enumext } { unknown-key-family }
3318 {
3319   Unknown~key~family~`\l_keys_key_str'~for~enumext.
3320 }
```

Messages used in length calculation.

```

3321 \msg_new:nnn { enumext } { width-negative }
3322 {
3323   Ignoring ~ negative ~ value ~ '#1=#2' ~ \msg_line_context:.\
3324   The ~ key ~ '#1'~ accepts ~ values ~ >= ~ 0pt.
3325 }
3326 \msg_new:nnn { enumext } { width-zero }
3327 {
3328   Invalid ~ '#1=#2' ~ \msg_line_context:.\
3329   The ~ key ~ '#1'~ accepts ~ values ~ > ~ 0pt.
3330 }
```

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

```

3331 \msg_new:nnn { enumext } { list-lengths }
3332 {
3333   **** ~ Lengths ~ used ~ by ~ 'enumext' ~ level ~ '#2' ~ \msg_line_context:~\c_space_tl ****\
3334   \__enumext_show_length:nnn { dim } { labelsep } { #1}
3335   \__enumext_show_length:nnn { dim } { labelwidth } { #1}
3336   \__enumext_show_length:nnn { dim } { itemindent } { #1}
3337   \__enumext_show_length:nnn { dim } { leftmargin } { #1}
3338   \__enumext_show_length:nnn { dim } { rightmargin } { #1}
3339   \__enumext_show_length:nnn { dim } { listparindent } { #1}
3340   \__enumext_show_length:nnn { skip } { topsep } { #1}
3341   \__enumext_show_length:nnn { skip } { parsep } { #1}
3342   \__enumext_show_length:nnn { skip } { partopsep } { #1}
3343   \__enumext_show_length:nnn { skip } { itemsep } { #1}
3344   ****~
3345 }
```

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

```

3346 \msg_new:nnn { enumext } { list-lengths-not-nested }
3347 {
3348   **** ~ Lengths ~ used ~ by ~ '#2' ~ environment ~ \msg_line_context:~\c_space_tl ****\
3349   \__enumext_show_length:nnn { dim } { labelsep } { #1}
3350   \__enumext_show_length:nnn { dim } { labelwidth } { #1}
3351   \__enumext_show_length:nnn { dim } { itemindent } { #1}
3352   \__enumext_show_length:nnn { dim } { leftmargin } { #1}
```

```

3353     \__enumext_show_length:nnn { dim } { rightmargin } {#1}
3354     \__enumext_show_length:nnn { dim } { listparindent } {#1}
3355     \__enumext_show_length:nnn { skip } { topsep } {#1}
3356     \__enumext_show_length:nnn { skip } { parsep } {#1}
3357     \__enumext_show_length:nnn { skip } { partopsep } {#1}
3358     \__enumext_show_length:nnn { skip } { itemsep } {#1}
3359     *****
3360 }

```

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

```

3361 \msg_new:nnn { enumext } { items-same-answer }
3362 {
3363     *****~Checking~answers~on~'#1'~OK~*****\\
3364     **~ All ~ items ~ stored ~ in ~ sequence ~ '#1' ~ have ~ an ~ answer. \\
3365     *****
3366     \prg_replicate:nn { 7 + \str_count:n {#1} } { * }
3367 }
3368 \msg_new:nnn { enumext } { item-different-answer }
3369 {
3370     Number ~ of ~ items ~ different ~ of ~ number ~ of ~
3371     answer ~ in ~ sequence ~ '#1'~ closed ~ \msg_line_context:.
3372 }

```

Messages used by the internal system to check for “starred” `\item*` commands.

```

3373 \msg_new:nnn { enumext } { missing-starred }
3374 {
3375     Missing ~ '\c_backslash_str #1*' ~ in ~ '#2' ~ \msg_line_context:.
3376 }

```

Message for the nesting depth of the environment `enumext`.

```

3377 \msg_new:nnn { enumext } { list-too-deep }
3378 {
3379     Too ~ deep ~ nesting ~ for ~ 'enumext' ~ \msg_line_context:~ \\
3380     The ~ maximum ~ level ~ of ~ nesting ~ is ~ 4.
3381 }

```

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

```

3382 \msg_new:nnn { enumext } { anskey-wrong-place }
3383 {
3384     Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:~ \\
3385     '\c_backslash_str #1' ~ works ~ in ~ the ~ environment ~ '#2'.
3386 }
3387 \msg_new:nnn { enumext } { anspic-wrong-place }
3388 {
3389     Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:~ \\
3390     '\c_backslash_str #1' ~ works ~ in ~ the ~ environment ~ '#2'.
3391 }
3392 \msg_new:nnn { enumext } { command-wrong-place }
3393 {
3394     Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:~ \\
3395     '\c_backslash_str #1' ~ works ~ outside ~ the ~ environment ~ '#2'.
3396 }

```

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

```

3397 \msg_new:nnn { enumext } { keyans-nested }
3398 {
3399     The ~ environment ~ 'keyans' ~ can't ~ be ~ nested ~ \msg_line_context:.
3400 }
3401 \msg_new:nnn { enumext } { keyans-wrong-level }
3402 {
3403     Wrong ~ level ~ position ~ for ~ 'keyans' ~ \msg_line_context:~ \\
3404     The ~ environment ~ 'keyans' ~ can ~ only ~ be ~ in ~ the ~ first ~ level.
3405 }
3406 \msg_new:nnn { enumext } { wrong-place }
3407 {
3408     Wrong ~ place ~ for ~ '#1' ~ environment ~ \msg_line_context:~ \\
3409     '#1' ~ is ~ only ~ found ~ with ~ '#2' ~ in ~ 'enumext.
3410 }
3411 \msg_new:nnn { enumext } { keyanspic-nested }
3412 {
3413     The ~ environment ~ 'keyanspic' ~ can't ~ be ~ nested~ \msg_line_context:~.
3414 }
3415 \msg_new:nnn { enumext } { keyanspic-wrong-level }

```

```

3416 {
3417     Wrong ~ level ~ position ~ for ~ 'keyanspic' ~ \msg_line_context:~ \\
3418     The ~ environment ~ 'keyans' ~ can ~ only ~ be ~ in ~ the ~ first ~ level.
3419 }

```

Messages used by `\getkeyans` command.

```

3420 \msg_new:nnn { enumext } { undefined-storage-anskey }
3421 {
3422     Storage ~ named ~ '#1' ~ is ~ not ~ defined ~ \msg_line_context:.
3423 }

```

Messages used by `\miniright` command.

```

3424 \msg_new:nnn { enumext } { missing-miniright }
3425 {
3426     Missing ~ '\c_backslash_str miniright' ~ in ~ \msg_line_context:.\
3427     The ~ key ~ 'mini-env' ~ need ~ '\c_backslash_str miniright'.
3428 }
3429 \msg_new:nnn { enumext } { wrong-miniright-place }
3430 {
3431     Wrong ~ place ~ for ~ '\c_backslash_str miniright' ~ \msg_line_context:~ \\
3432     Works ~ in ~ 'enumext' ~ and ~ 'keyans' ~ with ~ key ~ 'mini-env'.
3433 }
3434 \msg_new:nnn { enumext } { wrong-miniright-use }
3435 {
3436     Wrong ~ use ~ for ~ '\c_backslash_str miniright' ~ \msg_line_context:~ \\
3437     '\c_backslash_str miniright' ~ need ~ a ~ key ~ 'mini-env'.
3438 }

```

Messages used by `enumext*` and `keyans*` environments.

```

3439 \msg_new:nnn { enumext } { nested }
3440 {
3441     The ~ starred ~ environment ~ can't ~ be ~ nested ~ \msg_line_context:.
3442 }
3443 \msg_new:nnn { enumext } { item-joined }
3444 {
3445     Items ~ joined ~ (#1) ~ > ~ #2 ~ columns ~ \msg_line_context:.
3446 }
3447 \msg_new:nnn { enumext } { item-joined-columns }
3448 {
3449     Not ~ space ~ to ~ join ~ items ~ (#1) ~ > ~ #2 ~ \msg_line_context:.
3450 }

```

10.40 Finish package

Finish package implementation.

```

3451 \file_input_stop:
3452 \endpackage

```

11 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.

Symbols	
<code>*</code>	391
<code>\+</code>	200
<code>\-</code>	200
<code>\\</code> 208, 2621, 3310, 3323, 3328, 3333, 3348, 3363, 3364, 3379, 3384, 3389, 3394, 3403, 3408, 3417, 3426, 3431, 3436	
A	
above	<u>1202</u>
above*	<u>1202</u>
<code>\addvspace</code> .. 849, 877, 1000, 1079, 1142, 1148, 1176, 1193, 2440, 2460, 2577, 2592, 2816, 2823	
after	687
align	<u>345</u>
<code>\Alph</code>	29, 33
<code>\Alph</code>	297, 474, 492, 505, 3187
<code>\alph</code>	29, 33
<code>\alph</code>	298, 472, 3177
<code>\anskey</code>	10, 56, <u>1641</u>
<code>\anspic</code>	12, 76, 77, <u>2599</u>
<code>\arabic</code>	29, 30
<code>\arabic</code>	296, 471, 491, 3172, 3192
B	
<code>\b</code>	2331, 2344, 2885, 2898
<code>\baselineskip</code>	41
<code>\baselineskip</code>	1601, 1609
before	687
before*	687
below	<u>1202</u>
below*	<u>1202</u>
bool commands:	
<code>\bool_gset_false:N</code>	2483, 2825, 2934, 3137
<code>\bool_gset_true:N</code> 791, 2315, 2445, 2808, 2826, 2867, 2931	
<code>\bool_if:NTF</code> . 237, 249, 266, 1224, 1238, 1251, 1262, 1273, 1284, 1295, 1306, 1337, 1344, 1355, 1413, 1415, 1563, 1587, 1594, 1622, 1653, 1666, 1668, 1679, 1699, 1824, 1835, 1839, 1901, 1935, 1950, 1961, 2038, 2069, 2143, 2159, 2227, 2237, 2273, 2278, 2322, 2329, 2342, 2374, 2424, 2438, 2451, 2476, 2506, 2562, 2575, 2583, 2601, 2804, 2813, 2817, 2875, 2883, 2896, 2939, 2949, 3032, 3038, 3041, 3055, 3059, 3074, 3103, 3130	
<code>\bool_if:nTF</code> 1177, 1194, 1707, 2081, 2115, 2179, 2622, 3211	
<code>\bool_if_p:N</code>	2216, 2263
<code>\bool_lazy_all:nTF</code> .. 1393, 1756, 1765, 1778, 1794, 2309, 2861	
<code>\bool_lazy_and:nnTF</code> . 1689, 1732, 1920, 2214, 2262, 2358, 2443, 2929	
<code>\bool_lazy_or:nnTF</code>	2627
<code>\bool_new:N</code> 25, 26, 27, 28, 29, 35, 37, 46, 67, 72, 73, 78, 79, 82, 98, 100, 102, 105, 106, 115, 116, 117, 128, 129, 154, 165, 167	
<code>\bool_not_p:n</code> 1691, 1783, 1798, 2311, 2358, 2443, 2863	
<code>\bool_set_eq:NN</code>	2047, 2096, 2982
<code>\bool_set_false:N</code> 246, 1488, 1491, 1511, 1514, 2465, 2513, 2595, 2659, 2676, 2935, 2979	
<code>\bool_set_true:N</code> 228, 232, 338, 615, 1208, 1213, 1332, 1351, 1362, 1487, 1490, 1510, 1513, 1523, 1530, 2043, 2074, 2092, 2104, 2308, 2362, 2367, 2393, 2511, 2537, 2793, 2860, 2988, 2995, 2996	
box commands:	
<code>\box_dp:N</code> . 896, 900, 904, 915, 919, 930, 939, 945, 955, 968, 974, 980, 1011, 1012, 1013, 1016, 1026, 1030, 1039, 1046, 1051, 1059, 1088, 1089, 1092, 1099, 1112, 1120, 1126, 1134, 2688	
<code>\box_new:N</code>	43, 160
<code>\box_set_wd:Nn</code>	3095
<code>\box_use:N</code>	3102
<code>\box_wd:N</code>	304
C	
<code>\c</code>	391, 392, 515, 517, 529, 531
<code>\cB</code>	392
<code>\cE</code>	392
<code>\centering</code>	1179, 1196, 2709, 2819
check-ans	<u>1364</u>
Document class:	
article	34
clist commands:	
<code>\clist_const:Nn</code>	172
<code>\clist_map_function:nN</code>	2696
<code>\clist_map_inline:Nn</code> . 344, 557, 620, 686, 701, 782, 1218	
<code>\clist_map_inline:nn</code> . 34, 51, 57, 69, 81, 104, 127, 137, 151, 171, 369, 386, 625, 797, 1374, 1500, 1518, 1539, 1753, 1880, 1996, 2208, 2211, 2244, 2254, 2257, 2283	
<code>\columnbreak</code>	57
<code>\columnbreak</code>	1693
columns	<u>766</u>
columns*	<u>1519</u>
columns-sep	<u>766</u>
columns-sep*	<u>1519</u>
<code>\columnsep</code>	72, 75
<code>\columnsep</code>	2418, 2559
<code>\columnseprule</code>	72, 75
<code>\columnseprule</code>	2422, 2561
Commands provide by enumext :	
<code>\anskey</code> 23, 24, 49–51, 54, 56, 58–60, 62, 71, 84, 88, 89, 92	
<code>\anspic*</code>	23, 60–62, 77–79, 88, 89
<code>\anspic</code>	54, 76–79, 92
<code>\getkeyans</code>	54, 88, 93
<code>\item*</code>	23, 54, 60–62, 64, 65, 85, 88, 89
<code>\itemwidth</code>	80
<code>\item</code>	64, 65, 80, 84–86
<code>\miniright</code>	22, 38, 46, 47, 72, 73, 75, 76, 93
<code>\printkeyans</code>	23, 54, 88
<code>\setenumext</code>	23, 89–91
Counters defined by enumext :	
enumXiii	22, 28
enumXii	22, 28
enumXiv	22, 28
enumXi	22, 28
enumXviii	22, 28
enumXvii	22, 28, 86
enumXvi	22, 28
enumXv	22, 28

cs commands:

`\cs_generate_variant:Nn` 306, 322, 521, 537, 1553, 1560, 1640, 2198, 2698
`\cs_if_exist:NTF` 276
`\cs_new:Nn` 185
`\cs_new:Npn` 189, 194, 204
`\cs_new_eq:NN` 212, 213, 214, 218, 219, 251, 252, 255, 256
`\cs_new_protected:Nn` . 223, 387, 407, 439, 702, 706, 710, 714, 718, 722, 726, 730, 734, 738, 742, 746, 750, 754, 758, 762, 798, 810, 834, 851, 862, 886, 961, 985, 1002, 1064, 1081, 1103, 1138, 1144, 1219, 1233, 1247, 1258, 1269, 1280, 1291, 1302, 1342, 1353, 1411, 1423, 1446, 1561, 1585, 1592, 1620, 1627, 1744, 1871, 2001, 2005, 2024, 2077, 2111, 2127, 2137, 2153, 2303, 2356, 2372, 2379, 2402, 2432, 2449, 2504, 2527, 2545, 2570, 2581, 2618, 2661, 2674, 2694, 2699, 2715, 2783, 2802, 2853, 2910, 2916, 2937, 2947, 2964, 3114
`\cs_new_protected:Npn` 177, 181, 259, 274, 291, 301, 307, 395, 414, 508, 522, 1166, 1185, 1329, 1379, 1544, 1554, 1676, 1821, 1833, 1855, 1906, 1940, 1948, 2034, 2053, 2088, 2100, 2167, 2201, 2247, 2318, 2327, 2523, 2669, 2734, 2870, 2881, 2970, 2977, 2993, 3001, 3006, 3018, 3145, 3158, 3205, 3282, 3294
`\cs_new_protected_nopar:Nn` 2957, 3090
`\cs_new_protected_nopar:Npn` 3024
`\cs_set:Nn` 192, 1826
`\cs_set:Npn` 1754, 1792, 3151
`\cs_set_eq:NN` 191, 196, 2843, 2844, 3026
`\cs_set_protected:Nn` 626, 642, 654, 666
`\cs_set_protected:Npn` . 30, 44, 52, 64, 70, 94, 122, 133, 145, 152, 323, 345, 374, 455, 475, 538, 558, 602, 621, 678, 687, 766, 783, 1202, 1364, 1472, 1501, 1519, 1746, 1873, 1985, 2199, 2245
`\cs_to_str:N` 293, 316

D

`\d` 200
`\DeclareDocumentEnvironment` 879

dim commands:

`\dim_abs:n` 2172, 2177
`\dim_add:Nn` 2679
`\dim_compare:nNnTF` . 628, 644, 656, 668, 1168, 1187, 2169, 2174, 2180, 2186, 2188, 2190, 2384, 2407, 2531, 2549, 2671, 2717, 2785
`\dim_compare:nTF` 1717
`\dim_gset_eq:NN` 2794
`\dim_gzero:N` 2828
`\dim_new:N` . 40, 47, 48, 49, 66, 101, 111, 161, 162, 168
`\dim_set:Nn` . . 304, 616, 1531, 2067, 2172, 2177, 2179, 2182, 2183, 2187, 2189, 2192, 2193, 2195, 2387, 2410, 2533, 2551, 2701, 2719, 2726, 2769, 2787, 3020
`\dim_set_eq:NN` 462, 482, 498, 502, 2062, 2210, 2256, 2346, 2418, 2559, 2776, 2779, 2780, 2900, 3011
`\dim_use:N` 629, 637, 1169, 1175, 1630, 1633, 1638, 2132, 2134, 2385, 2390, 2391, 2398, 2408, 2412, 2413, 2415
`\dim_zero:N` 2422, 2561, 2680, 2681, 2682
`\dim_zero_new:N` 2732
`\c_zero_dim` 631, 645, 657, 669, 1169, 1187, 1719, 2169, 2174, 2180, 2187, 2385, 2408, 2531, 2549, 2717, 2785

E

`\end` . . 1172, 1190, 1589, 1624, 2437, 2459, 2574, 2591, 2806, 2822, 3215, 3220
`\endlist` 26
`\endlist` 213

`\endlrbox` 3093
`\endminipage` 27
`\endminipage` 219
enumext 4, 2284
enumext internal commands:

`__enumext_add_pre_parsep:` . . . 39, 808, 810, 810
`__enumext_after_args_exec:` . 37, 702, 714, 2296
`__enumext_after_args_exec_v:` . 37, 38, 718, 730, 2497
`__enumext_after_args_exec_vii:` . . . 734, 758
`__enumext_after_args_exec_viii:` 762
`__enumext_after_env:n` 73
`__enumext_after_env:nn` . . 73, 87, 181, 181, 2474, 2811, 3128
`__enumext_after_hyperref:` . . . 27, 221, 223, 223
`__enumext_after_list:` . 73, 83, 2301, 2449, 2449
`\l__enumext_after_list_args_v_tl` 732
`\l__enumext_after_list_args_vii_tl` 760, 3084
`\l__enumext_after_list_args_viii_tl` . . . 764
`__enumext_after_list_v:` . . 76, 2502, 2581, 2581
`__enumext_after_list_vii:` . . . 2851, 2916, 2916
`__enumext_after_star_env:nn` 81
`__enumext_after_stop_list:` . . . 37, 38, 702, 710, 2463
`__enumext_after_stop_list_v:` 37, 718, 726, 2596
`\l__enumext_after_stop_list_v_tl` 728
`__enumext_after_stop_list_vii:` 734, 750, 2920
`\l__enumext_after_stop_list_vii_tl` . . . 752
`__enumext_after_stop_list_viii:` 754
`\l__enumext_after_stop_list_viii_tl` . . . 756
`\l__enumext_align_label_vii_str` . . 3076, 3080
`\l__enumext_align_label_X_str` 152
`\c__enumext_all_envs_clist` . . 172, 344, 557, 620, 686, 701, 782, 1218
`\c__enumext_all_families_seq` . . 90, 3244, 3253, 3275
`__enumext_anskey_wrapper:n` 1476, 1831
`__enumext_at_begin_document:n` . . 26, 177, 177, 210, 216
`__enumext_before_args_exec:` 36, 702, 702, 2382
`__enumext_before_args_exec_v:` . . 37, 718, 718, 2530
`__enumext_before_args_exec_vii:` . . 734, 734, 2913
`__enumext_before_args_exec_viii:` 738
`__enumext_before_keys_exec:` 37, 702, 706, 2294
`__enumext_before_keys_exec_v:` . . 37, 718, 722, 2495
`__enumext_before_keys_exec_vii` 734
`__enumext_before_keys_exec_vii:` 38, 742, 2839
`__enumext_before_keys_exec_viii:` . . 38, 746
`__enumext_before_list:` . . . 71, 2288, 2379, 2379
`__enumext_before_list_v:` . 75, 2490, 2527, 2527
`__enumext_before_list_vii:` 83, 2834, 2910, 2910
`\l__enumext_before_no_starred_key_v_tl` 724
`\l__enumext_before_no_starred_key_vii-`
`tl` 744
`\l__enumext_before_no_starred_key_viii-`
`tl` 748
`\l__enumext_before_starred_key_v_tl` . . . 720
`\l__enumext_before_starred_key_vii_tl` . 736
`\l__enumext_before_starred_key_viii_tl` 740
`__enumext_calc_hspace:NNNNNNN` 67, 2167, 2167, 2198, 2203, 2249


```

\__enumext_check_ans_active: . . 52, 1423, 1423,
    2480
\__enumext_check_ans_active_vii: . 1423, 1446,
    3134
\l__enumext_check_ans_bool . . . 49, 64, 115, 1337,
    1368, 1413, 1668, 1935, 2038, 2069, 2443, 2929, 3038
\__enumext_check_ans_count: . 51, 71, 1411, 1411,
    2383
\__enumext_check_ans_int:n . . 49, 51, 1339, 1379,
    1379
\g__enumext_check_ans_item_tl . . 62, 115, 1934,
    1942, 1946
\g__enumext_check_ans_show_bool 73, 115, 2445,
    2476, 2483
\g__enumext_check_ans_show_h_bool 115, 2931,
    3130, 3137
\l__enumext_columns_sep_v_dim 2549, 2551, 2559
\l__enumext_columns_sep_vii_dim . . 2717, 2719,
    2728, 2773, 2902, 3112
\l__enumext_columns_v_int 1007, 2547, 2555, 2567,
    2572
\l__enumext_columns_vii_int . . 2722, 2725, 2729,
    2737, 2741, 2744, 2750, 2756, 2760, 2889, 3107, 3118
\l__enumext_compare_items_ans_int 115, 1425,
    1431, 1448, 1457
\g__enumext_count_item_all_int 115, 1399, 1402,
    1427, 1450, 2040, 2071, 3047
\g__enumext_count_item_ans_int 51, 56, 62, 115,
    1409, 1431, 1457, 1670, 1937
\g__enumext_count_item_i_int . . . . 1404, 1451
\g__enumext_count_item_ii_int 1405, 1427, 1452
\g__enumext_count_item_iii_int 1406, 1428, 1453
\g__enumext_count_item_iv_int 1407, 1428, 1454
\g__enumext_count_item_vii_int . . . . . 1408
\g__enumext_count_item_X_int . . . . . 115
\g__enumext_count_level_i_int . . . 1440, 1466
\g__enumext_count_level_ii_int . . 1441, 1467
\g__enumext_count_level_iii_int . . 1442, 1468
\g__enumext_count_level_iv_int . . 1443, 1469
\g__enumext_count_level_vii_int . . 1444, 1470,
    3048
\g__enumext_count_level_X_int . . . . . 115
\l__enumext_counter_i_tl . . . . . 30, 283
\l__enumext_counter_ii_tl . . . . . 30, 284
\l__enumext_counter_iii_tl . . . . . 30, 285
\l__enumext_counter_iv_tl . . . . . 30, 286
\l__enumext_counter_style_for_ref_vii_-
    tl . . . . . 422, 432, 443, 445
\l__enumext_counter_style_for_ref_viii_-
    tl . . . . . 449, 451
\l__enumext_counter_style_for_ref_X_tl 141
\c__enumext_counter_style_tl . . . 30, 141, 389
\g__enumext_counter_styles_tl . 22, 29, 40, 294,
    312
\l__enumext_counter_v_tl . . . . . 30, 287
\l__enumext_counter_vi_tl . . . . . 30, 288
\l__enumext_counter_vii_tl . . . . 30, 289, 419
\l__enumext_counter_viii_tl . . . . 30, 290, 429
\l__enumext_current_widest_dim 22, 40, 318, 463,
    483, 499, 503
\__enumext_default_item:n . . . 2034, 2034, 2085
\__enumext_define_counters:Nn 22, 274, 274, 283,
    284, 285, 286, 287, 288, 289, 290
\__enumext_endminipage: . 27, 216, 219, 885, 2711,
    3092
\__enumext_fake_item: . . . . . 626, 626, 2236
\l__enumext_fake_item_indent_v_dim 645, 650
\l__enumext_fake_item_indent_v_tl 647, 2093,
    2097, 2105
\l__enumext_fake_item_indent_vii_dim 657, 662
\l__enumext_fake_item_indent_vii_tl 659, 3088
\l__enumext_fake_item_indent_viii_dim . 669,
    674
\l__enumext_fake_item_indent_viii_tl . . 671
\l__enumext_fake_item_indent_X_tl . . . . 70
\__enumext_fake_item_vii: . . . 626, 654, 2272
\__enumext_fake_item_viii: . . . 626, 666, 2277
\g__enumext_footnote_arg_seq . 138, 2007, 2020,
    2030
\g__enumext_footnote_int . 138, 2014, 2017, 2019,
    2021
\g__enumext_footnote_int_seq . 138, 2008, 2021,
    2026, 2029
\__enumext_footnotes_key_bool . . . . . 27
\l__enumext_footnotes_key_bool 24, 27, 86, 128,
    232, 237, 246, 3055, 3103
\__enumext_footnotetext:nn . . . 2001, 2001, 2031
\__enumext_getkeyans:nn . . . 88, 3154, 3158, 3158
\__enumext_getkeyans_aux:n . 88, 3142, 3145, 3145
\l__enumext_hyperref_bool 24, 27, 128, 228, 249,
    266, 1734, 1922, 3032
\__enumext_hypertarget:nn 27, 223, 251, 255, 271
\__enumext_if_is_int:n . . . . . 198
\__enumext_if_is_int:nTF . . . . . 198, 510, 524
\l__enumext_item_column_pos_vii_int 84, 2744,
    2750, 2756, 2760, 2767, 2960, 3107, 3110
\l__enumext_item_column_pos_X_int . . . . 152
\g__enumext_item_count_all_vii_int 84, 2768,
    2961, 3118, 3125
\g__enumext_item_count_all_X_int . . . . 152
\__enumext_item_peek_args_vii: 84, 2962, 2964,
    2964
\__enumext_item_starred: . . 66, 2127, 2127, 2145
\l__enumext_item_starred_vii_bool 2979, 2995,
    3059
\l__enumext_item_starred_X_bool . . . . 152
\__enumext_item_std:w 26, 64, 65, 79, 210, 214, 2044,
    2050, 2075, 2093, 2097, 2105, 2692
\g__enumext_item_symbol_aux_vii_tl 3003, 3061,
    3064, 3068, 3070
\g__enumext_item_symbol_aux_X_tl . . . . 152
\l__enumext_item_symbol_sep_vii_dim . . 3012,
    3020, 3067, 3069
\g__enumext_item_symbol_tl 22, 64, 35, 2059, 2133,
    2150
\l__enumext_item_symbol_vii_tl . . . . . 3064
\l__enumext_item_text_vii_box 3054, 3095, 3102
\l__enumext_item_text_X_box . . . . . 152
\l__enumext_item_width_vii_dim . . 2726, 2771,
    2779, 2780
\l__enumext_item_width_X_dim . . . . . 152
\l__enumext_itemindent_X_dim . . . . . 44
\l__enumext_itemsep_vii_skip . . . . . 3124
\l__enumext_joined_item_aux_vii_int . . 2765,
    2766, 2767, 2768, 2774
\l__enumext_joined_item_aux_X_int . . . 152
\__enumext_joined_item_vii:w . . 84, 2967, 2968,
    2970, 2970

```

```

\l__enumext_joined_item_vii_int .. 2736, 2737,
    2740, 2742, 2748, 2753, 2758, 2763, 2765, 2771
\l__enumext_joined_item_X_int ..... 152
\l__enumext_joined_width_vii_dim . 2769, 2776,
    2779, 3085, 3097
\l__enumext_joined_width_X_dim ..... 152
\__enumext_keyans_addto_prop:n 60, 1855, 1855,
    2107, 2624
\__enumext_keyans_addto_seq:n . 61, 1906, 1906,
    2109, 2626
\__enumext_keyans_anspic_code:nnn . 77, 2615,
    2618, 2618
\__enumext_keyans_check_ans:nn 62, 1940, 1940,
    2500, 2656
\__enumext_keyans_default_item:n .. 65, 2088,
    2088, 2123
\l__enumext_keyans_env_bool 20, 2358, 2511, 2595
\__enumext_keyans_fake_item: .. 626, 642, 2226
\__enumext_keyans_internal_ref: 60, 1871, 1871,
    2108, 2625
\l__enumext_keyans_level_h_int ..... 20
\l__enumext_keyans_level_int .. 20, 1160, 1657,
    1970, 2510, 2514, 2609
\__enumext_keyans_make_label: 29, 67, 2153, 2153,
    2225
\__enumext_keyans_mini_addvspace: 44, 75, 1064,
    1064, 2539
\__enumext_keyans_mini_right_cmd:n 47, 1162,
    1185, 1185
\__enumext_keyans_mini_set_vskip: . 43, 1002,
    1002, 1066
\__enumext_keyans_multi_addvspace: . 75, 851,
    862, 2564
\__enumext_keyans_multi_set_vskip: . 40, 851,
    851, 864
\__enumext_keyans_multicols_start: 75, 2543,
    2545, 2545
\__enumext_keyans_multicols_stop: . 76, 1189,
    2570, 2570, 2594
\__enumext_keyans_parse_keys:n 2489, 2523, 2523
\l__enumext_keyans_pic_above_int . 110, 2702,
    2703, 2705
\l__enumext_keyans_pic_above_skip .. 79, 110,
    2647, 2686
\__enumext_keyans_pic_arg_two: 78, 2645, 2674,
    2674
\l__enumext_keyans_pic_below_int . 110, 2702,
    2703, 2706
\l__enumext_keyans_pic_body_seq .. 77-79, 110,
    2613, 2652, 2710
\__enumext_keyans_pic_do:n 79, 2652, 2654, 2694,
    2694, 2698
\l__enumext_keyans_pic_level_int .. 20, 1152,
    1661, 1858, 1881, 1909, 2663, 2664
\__enumext_keyans_pic_row:n 79, 2696, 2699, 2699
\__enumext_keyans_pic_safe_exec: .. 78, 2641,
    2661, 2661
\__enumext_keyans_pic_skip_abs:N .. 78, 2669,
    2669, 2685
\l__enumext_keyans_pic_width_dim . 110, 2701,
    2708
\__enumext_keyans_redefine_item: .. 66, 2111,
    2111, 2224
\__enumext_keyans_safe_exec: . 2488, 2504, 2504

\__enumext_keyans_show_left:n . 65, 1948, 1948,
    2103, 2632
\__enumext_keyans_starred_item:n .. 65, 2100,
    2100, 2119
\l__enumext_keyans_tmpa_tl .. 23, 82, 2102, 2106
\l__enumext_label_copy_i_tl .. 1788, 1884, 1888
\l__enumext_label_copy_v_tl ..... 1888
\l__enumext_label_copy_vi_tl ..... 1884
\l__enumext_label_copy_vii_tl 1763, 1774, 1805
\l__enumext_label_copy_X_tl ..... 130
\l__enumext_label_fill_left_v_tl ..... 2157
\l__enumext_label_fill_left_X_tl ..... 70
\l__enumext_label_fill_right_v_tl .... 2164
\l__enumext_label_fill_right_X_tl ..... 70
\l__enumext_label_font_style_v_tl 2158, 2636
\l__enumext_label_font_style_vii_tl ... 3073
\l__enumext_label_i_tl ..... 455
\l__enumext_label_ii_tl ..... 455
\l__enumext_label_iii_tl ..... 455
\l__enumext_label_iv_tl ..... 455
\__enumext_label_style:Nnn 22, 29, 307, 307, 322,
    460, 480, 496, 500
\l__enumext_label_v_tl .. 60, 61, 493, 1863, 1914,
    1952, 1959, 1975, 1982, 2102, 2106, 2492, 2631, 2633
\l__enumext_label_vi_tl . 60, 61, 493, 1860, 1911,
    2631, 2633, 2637
\l__enumext_label_vii_tl . 475, 2990, 3015, 3022
\l__enumext_label_viii_tl ..... 475
\l__enumext_label_width_by_box .. 40, 303, 304
\__enumext_label_width_by_box:Nn 29, 301, 301,
    306, 318, 534
\l__enumext_labelsep_i_dim ..... 1956, 1979
\l__enumext_labelsep_v_dim ..... 2554
\l__enumext_labelsep_vii_dim . 2721, 2730, 2772,
    3013, 3083, 3099
\l__enumext_labelwidth_i_dim ..... 1955, 1978
\l__enumext_labelwidth_v_dim ..... 2554
\l__enumext_labelwidth_vii_dim ... 2721, 2729,
    2772, 3076, 3080, 3098
\l__enumext_leftmargin_tmp_v_bool . 78, 2676
\l__enumext_leftmargin_tmp_X_bool ..... 44
\l__enumext_leftmargin_tmp_X_dim ..... 44
\l__enumext_leftmargin_X_dim ..... 44
\__enumext_level: 185, 185, 191, 192, 196, 398, 400,
    401, 409, 411, 629, 633, 637, 704, 708, 712, 716, 800,
    802, 804, 806, 839, 841, 843, 845, 849, 889, 892, 911,
    920, 926, 931, 935, 946, 950, 951, 956, 992, 996, 1169,
    1175, 1222, 1224, 1226, 1229, 1236, 1238, 1240, 1243,
    1417, 1418, 1420, 1565, 1573, 1577, 1581, 1826, 1829,
    1830, 2041, 2043, 2044, 2048, 2049, 2050, 2057, 2059,
    2063, 2064, 2067, 2072, 2074, 2075, 2129, 2132, 2134,
    2141, 2142, 2143, 2146, 2149, 2291, 2293, 2329, 2334,
    2335, 2336, 2338, 2342, 2347, 2348, 2349, 2351, 2362,
    2367, 2374, 2385, 2387, 2390, 2391, 2393, 2398, 2405,
    2408, 2410, 2412, 2413, 2414, 2415, 2418, 2424, 2429,
    2435, 2438, 2440, 2451, 3043, 3044
\__enumext_level_ ..... 191
\__enumext_level_end:n ..... 189, 194
\l__enumext_level_h_int 20, 417, 441, 1782, 1799,
    2312, 2360, 2855, 2856
\l__enumext_level_int 20, 187, 812, 963, 1156, 1396,
    1759, 1769, 1775, 1781, 1789, 1797, 1804, 2239, 2305,
    2306, 2321, 2365, 2420, 2478, 2518, 2605, 2864, 2941,
    2951, 3132

```

<code>__enumext_level_set:n</code>	189 , 189
<code>__enumext_list_arg_two_i:</code>	2199
<code>__enumext_list_arg_two_ii:</code>	2199
<code>__enumext_list_arg_two_iii:</code>	2199
<code>__enumext_list_arg_two_iv:</code>	2199
<code>__enumext_list_arg_two_v:</code>	66 , 2199 , 2494 , 2677
<code>__enumext_list_arg_two_vii:</code>	2245 , 2838
<code>__enumext_list_arg_two_viii:</code>	2245
<code>\l__enumext_listoffset_v_dim</code>	2556
<code>\l__enumext_listparindent_vii_dim</code>	3086
<code>__enumext_make_label:</code> 29 , 64 , 66 , 2137 , 2137 , 2234	
<code>\l__enumext_mark_answer_sym_tl</code>	56 , 62 , 105 , 1479 , 1635 , 1841 , 1963
<code>\l__enumext_mark_position_str</code> 105 , 1483 , 1484 ,	1506 , 1507 , 1633
<code>\l__enumext_mark_ref_sym_tl</code>	105 , 1493 , 1739 , 1930
<code>__enumext_mini_addvspace:</code> 43 , 72 , 985 , 985 , 2395	
<code>__enumext_mini_addvspace_vii:</code> 45 , 1138 , 1138 ,	2797
<code>__enumext_mini_addvspace_viii:</code> 45 , 1138 , 1144	
<code>__enumext_mini_env*</code>	879
<code>__enumext_mini_right_cmd:n</code>	46 , 47 , 1164 , 1166 , 1166
<code>__enumext_mini_set_vskip:</code>	41 , 886 , 886 , 987
<code>__enumext_mini_set_vskip_vii:</code> 45 , 1081 , 1081 ,	1140
<code>__enumext_mini_set_vskip_viii:</code> 45 , 1081 , 1103 ,	1146
<code>__enumext_minipage:w</code> 27 , 216 , 218 , 881 , 2708 , 3085	
<code>\l__enumext_minipage_active_v_bool</code>	75 , 76 , 2537 , 2562 , 2575 , 2583
<code>\g__enumext_minipage_active_vii_bool</code>	81 , 2808 , 2813 , 2825
<code>\l__enumext_minipage_active_vii_bool</code>	2793 , 2804
<code>\g__enumext_minipage_active_X_bool</code>	152
<code>\l__enumext_minipage_active_X_bool</code>	58
<code>\g__enumext_minipage_after_skip</code> 58 , 1085 , 1097 ,	2823
<code>\l__enumext_minipage_after_skip</code> 41 , 42 , 73 , 76 ,	58 , 902 , 917 , 937 , 953 , 968 , 974 , 980 , 994 , 1004 , 1013 , 1016 , 1028 , 1046 , 1057 , 1073 , 1105 , 1118 , 1132 , 2460 , 2592
<code>\g__enumext_minipage_center_vii_bool</code>	2817 , 2826
<code>\g__enumext_minipage_center_X_bool</code>	152
<code>\l__enumext_minipage_hsep_v_dim</code>	75 , 2535
<code>\l__enumext_minipage_hsep_vii_dim</code>	2791
<code>\l__enumext_minipage_left_skip</code> 41 , 75 , 58 , 894 ,	909 , 928 , 943 , 990 , 1000 , 1005 , 1011 , 1020 , 1037 , 1049 , 1069 , 1079 , 1083 , 1088 , 1092 , 1106 , 1110 , 1124 , 1142 , 1148
<code>\l__enumext_minipage_left_v_dim</code> 75 , 2533 , 2541	
<code>\l__enumext_minipage_left_vii_dim</code> 2787 , 2799	
<code>\l__enumext_minipage_left_X_dim</code>	58
<code>\g__enumext_minipage_right_skip</code> 58 , 1084 , 1089 ,	1093 , 2816
<code>\l__enumext_minipage_right_skip</code>	41 , 58 , 898 , 913 , 933 , 948 , 1006 , 1012 , 1024 , 1042 , 1053 , 1107 , 1114 , 1128 , 1176 , 1193
<code>\l__enumext_minipage_right_v_dim</code>	75 , 1187 , 1192 , 2531 , 2535
<code>\g__enumext_minipage_right_vii_dim</code> 81 , 2795 ,	2815 , 2828
<code>\l__enumext_minipage_right_vii_dim</code> 81 , 2785 ,	2790 , 2796
<code>\g__enumext_minipage_right_X_dim</code>	152
<code>\g__enumext_minipage_right_X_skip</code>	152
<code>\g__enumext_minipage_stat_int</code>	72 , 75 , 58 , 1181 , 1198 , 2394 , 2453 , 2458 , 2538 , 2585 , 2590
<code>\g__enumext_miniright_code_vii_tl</code>	81 , 2821 , 2827
<code>\g__enumext_miniright_code_X_tl</code>	152
<code>__enumext_multi_addvspace:</code>	40 , 72 , 834 , 834 , 2426
<code>__enumext_multi_set_vskip:</code>	39 , 798 , 798 , 836
<code>\l__enumext_multicols_above_ii_skip</code>	817
<code>\l__enumext_multicols_above_iii_skip</code>	823
<code>\l__enumext_multicols_above_iv_skip</code>	829
<code>\l__enumext_multicols_above_v_skip</code> 853 , 867 ,	877
<code>\l__enumext_multicols_above_X_skip</code>	52
<code>\l__enumext_multicols_below_v_skip</code> 857 , 871 ,	2577
<code>\l__enumext_multicols_below_X_skip</code>	52
<code>__enumext_multicols_start:</code> 72 , 2400 , 2402 , 2402	
<code>__enumext_multicols_stop:</code> 73 , 1171 , 2432 , 2432 ,	2462
<code>__enumext_newlabel:nn</code>	24 , 27 , 59 , 61 , 259 , 259 , 1815 , 1897
<code>\l__enumext_newlabel_arg_one_tl</code> 24 , 27 , 59 , 61 ,	130 , 1738 , 1808 , 1816 , 1890 , 1898 , 1928
<code>\l__enumext_newlabel_arg_two_tl</code> 24 , 28 , 58 , 60 ,	130 , 1762 , 1772 , 1786 , 1802 , 1817 , 1883 , 1887 , 1899
<code>__enumext_parse_keys:n</code>	2287 , 2318 , 2318
<code>__enumext_parse_keys_vii:n</code>	2833 , 2870 , 2870
<code>__enumext_parse_store_keys:n</code>	70 , 2324 , 2327 , 2327
<code>__enumext_parse_store_keys_vii:n</code> 82 , 83 , 2877 ,	2881 , 2881
<code>\l__enumext_parsep_i_skip</code>	815 , 817 , 966 , 1014
<code>\l__enumext_parsep_ii_skip</code>	821 , 823 , 972
<code>\l__enumext_parsep_iii_skip</code>	827 , 829 , 978
<code>\l__enumext_parsep_vii_skip</code>	3087
<code>\l__enumext_partopsep_v_skip</code>	869 , 873 , 1040 , 1044 , 1051 , 1055 , 1071 , 1075
<code>\l__enumext_partopsep_viii_skip</code>	1116
<code>__enumext_phantomsection:</code> 27 , 223 , 252 , 256 , 272	
<code>__enumext_print_footnote:</code>	2001 , 2024 , 3105
<code>__enumext_print_keyans_box:NN</code> 56 , 1627 , 1627 ,	1640 , 1828 , 1954 , 1977
<code>\l__enumext_print_keyans_i_tl</code>	3168 , 3197
<code>\l__enumext_print_keyans_ii_tl</code>	3173 , 3198
<code>\l__enumext_print_keyans_iii_tl</code>	3178 , 3199
<code>\l__enumext_print_keyans_iv_tl</code>	3183 , 3200
<code>\l__enumext_print_keyans_vii_tl</code>	3188 , 3201
<code>\l__enumext_print_keyans_X_tl</code>	94
<code>__enumext_printkeyans:nnn</code>	89 , 3202 , 3205 , 3205
<code>__enumext_redefine_item:</code>	65 , 2077 , 2077 , 2233
<code>\l__enumext_ref_aux_tl</code> 141 , 398 , 400 , 403 , 419 , 421 ,	424 , 429 , 431 , 434
<code>\l__enumext_ref_key_arg_tl</code>	141 , 392 , 397 , 404 , 416 , 425 , 435
<code>__enumext_regex_label_ref_key:</code>	30 , 31 , 387 , 387 , 399 , 420 , 430

```

\__enumext_register_counter_style:Nn . . . 291,
    291, 296, 297, 298, 299, 300
\__enumext_remove_extra_parsep_vii: . . . 2848,
    3114, 3114
\__enumext_renew_footnote: . . . 2001, 2005, 3057
\l__enumext_resume_bool . . . . . 22, 35, 1351, 2216
\__enumext_resume_counter: . . 50, 1317, 1342, 1342
\__enumext_resume_counter_star: . . . . . 1319
\__enumext_resume_counter_vii: . . 50, 1326, 1342,
    1353
\g__enumext_resume_int 22, 73, 35, 1346, 1357, 2217,
    2466
\l__enumext_resume_vii_bool . . . 35, 1362, 2263
\g__enumext_resume_vii_int . . . 83, 35, 2264, 2922
\__enumext_safe_exec: . . . . . 2286, 2303, 2303
\__enumext_safe_exec_vii: . . . 2832, 2853, 2853
\__enumext_set_error:nn . . . . . 3282, 3292, 3294
\__enumext_set_label_ref:n . . . 30, 395, 395, 467
\__enumext_set_label_ref_h:n . . 31, 414, 414, 487
\__enumext_set_parse:n . . . . . 3265, 3282, 3282
\l__enumext_setkey_tmpa_int . . . 89, 3258, 3262
\l__enumext_setkey_tmpa_seq 89, 3256, 3266, 3272,
    3274, 3276, 3289
\l__enumext_setkey_tmpa_tl . . . . 89, 3264, 3268
\l__enumext_setkey_tmpb_seq 89, 3257, 3260, 3264,
    3265
\l__enumext_setkey_tmpb_tl 89, 3284, 3286, 3287
\l__enumext_show_answer_bool . . . 105, 1487, 1491,
    1510, 1514, 1835, 1950, 2628
\__enumext_show_length:nnn . . . 36, 204, 204, 3334,
    3335, 3336, 3337, 3338, 3339, 3340, 3341, 3342, 3343,
    3349, 3350, 3351, 3352, 3353, 3354, 3355, 3356, 3357,
    3358
\l__enumext_show_position_bool 105, 1488, 1490,
    1511, 1513, 1839, 1961, 2629
\g__enumext_standar_bool . . . . . 20, 2315
\l__enumext_standar_bool . . . 20, 1767, 1780, 1796,
    2308, 2465, 2863
\__enumext_standard_item_vii:w . . . 84, 85, 2975,
    2977, 2977
\g__enumext_starred_bool . . . 82, 83, 20, 1395, 1758,
    1768, 1798, 2443, 2867, 2929, 2934
\l__enumext_starred_bool . . . 82, 83, 20, 1691, 1699,
    1783, 1824, 2311, 2860, 2935
\__enumext_starred_columns_set_vii: . . . 2715,
    2715, 2841
\__enumext_starred_item:nn . . . 2053, 2053, 2083
\__enumext_starred_item_vii:w 84, 85, 2974, 2993,
    2993
\__enumext_starred_item_vii_aux_i:w . . . 2993,
    2998, 3001
\__enumext_starred_item_vii_aux_ii:w . . . 2993,
    2999, 3004, 3006
\__enumext_starred_item_vii_aux_iii:w . . . 2993,
    3009, 3018
\__enumext_starred_joined_item_vii:n . . . 80, 84,
    2734, 2734, 2972
\__enumext_start_from:NNn 33, 508, 508, 521, 543
\__enumext_start_item_tmp_vii: . . . 82, 2844, 2957,
    2957
\__enumext_start_item_vii:w . . . 85, 86, 2985, 2990,
    3015, 3022, 3024, 3024
\__enumext_start_list:nn 26, 68, 78, 210, 212, 2290,
    2491, 2642, 2836
\__enumext_start_mini_vii: . . . 83, 2783, 2783, 2914
\__enumext_start_store_level: . . . 71, 2289, 2356,
    2356
\__enumext_start_store_level_vii: . . . 84, 2835,
    2937, 2937
\l__enumext_start_X_int . . . . . 70, 538
\__enumext_stop_item_tmp_vii: . . . 82, 84, 86, 2843,
    2847, 2959, 3026
\__enumext_stop_item_vii: 86, 87, 3026, 3090, 3090
\__enumext_stop_list: . . . 26, 210, 213, 2299, 2501,
    2655, 2849
\__enumext_stop_mini_vii: 81, 83, 2802, 2802, 2919
\__enumext_stop_store_level: . . . 71, 2300, 2356,
    2372
\__enumext_stop_store_level_vii: . . . 84, 2850,
    2937, 2947
\l__enumext_store_active_bool 23, 49, 70, 82, 82,
    1332, 1344, 1355, 1653, 2322, 2358, 2506, 2513, 2601,
    2659, 2875, 2939, 2949
\__enumext_store_addto_prop:n 54, 60, 1544, 1544,
    1553, 1678, 1869
\__enumext_store_addto_seq:n 54, 61, 1554, 1554,
    1560, 1567, 1581, 1589, 1598, 1616, 1624, 1742, 1933
\l__enumext_store_ans_bool . . . 1370, 1415, 1563,
    1587, 1594, 1622, 1666, 3041
\l__enumext_store_anskey_arg_tl . . . 23, 57, 82,
    1684, 1693, 1695, 1701, 1709, 1712, 1722, 1727, 1730,
    1736, 1742
\__enumext_store_anskey_code:nnnn 56, 57, 1672,
    1676, 1676
\__enumext_store_anskey_show_left:n 59, 1683,
    1833, 1833
\__enumext_store_anskey_show_wrap:n 59, 1821,
    1821, 1837, 1852
\l__enumext_store_columns_break_bool . . . 1647,
    1690
\l__enumext_store_columns_join_int 82, 1698,
    1703
\l__enumext_store_columns_sep_vii_bool 2896
\l__enumext_store_columns_sep_vii_dim 2901,
    2905
\l__enumext_store_columns_sep_X_bool . . . 94
\l__enumext_store_columns_sep_X_dim . . . . . 94
\l__enumext_store_columns_vii_bool . . . 2883
\l__enumext_store_columns_vii_int 2888, 2892
\l__enumext_store_columns_X_bool . . . . . 94
\l__enumext_store_columns_X_int . . . . . 94
\__enumext_store_internal_ref: 58, 1681, 1744,
    1744
\l__enumext_store_item_symbol_sep_dim 1645,
    1719, 1724
\l__enumext_store_item_symbol_tl . . . 1643, 1710,
    1714
\l__enumext_store_keyans_label_tl 23, 60–62,
    82, 1857, 1860, 1863, 1867, 1869, 1908, 1911, 1914,
    1918, 1924, 1933, 1934
\__enumext_store_level_close: . . . 54, 1561, 1585,
    2376
\__enumext_store_level_close_vii: 1592, 1620,
    2953
\__enumext_store_level_open: . . . 53, 54, 70, 1561,
    1561, 2363, 2368
\__enumext_store_level_open_vii: . . . 83, 1592,
    1592, 2943

```


<code>\g__enumext_store_name_tl</code>	23, 73, <u>82</u> , 1434, 1438, 1460, 1464, 2446, 2484, 2932, 3138
<code>\l__enumext_store_name_tl</code>	23, 49, <u>82</u> , 1331, 1348, 1359, 1546, 1547, 1548, 1550, 1556, 1557, 1558, 1810, 1811, 1847, 1892, 1893, 1969, 2446, 2467, 2470, 2923, 2926, 2932
<code>\l__enumext_store_opt_vii_tl</code>	. 1596, 1606, 1612, 1616, 2890, 2903
<code>\l__enumext_store_opt_X_tl</code> <u>94</u>
<code>\l__enumext_store_ref_key_bool</code>	57, 1496, 1679, 1733, 1901, 1921
<code>\l__enumext_store_upper_level_X_bool</code>	... <u>94</u>
<code>\l__enumext_store_write_aux_file_tl</code>	24, 59, 61, <u>130</u> , 1813, 1819, 1895, 1903
<code>__enumext_storing_set:n</code>	49, 50, 1315, 1324, <u>1329</u> , 1329
<code>\l__enumext_the_counter_vii_tl</code> 421
<code>\l__enumext_the_counter_viii_tl</code> 431
<code>\l__enumext_the_counter_X_tl</code> <u>141</u>
<code>__enumext_tmp:n</code>	30, 34, 44, 51, 52, 57, 64, 69, 70, 81, 94, 104, 122, 127, 133, 137, 145, 151, 152, 171, 621, 625, 1364, 1378, 1472, 1500, 1501, 1518, 1746, 1753, 1754, 1775, 1789, 1792, 1804, 1873, 1880, 2199, 2244, 2245, 2283
<code>__enumext_tmp:nn</code>	323, 344, 345, 373, 374, 386, 538, 557, 602, 620, 678, 686, 687, 701, 766, 782, 783, 797, 1202, 1218, 1519, 1543, 1985, 2000
<code>__enumext_tmp:nnn</code>	455, 471, 472, 473, 474, 475, 491, 492
<code>__enumext_tmp:nnnnn</code>	558, 583, 586, 589, 591, 593, 596, 599
<code>__enumext_tmp:w</code> 3151, 3154
<code>\l__enumext_tmpa_vii_int</code> 2725, 2728
<code>\l__enumext_tmpa_X_int</code> <u>152</u>
<code>\l__enumext_topsep_v_skip</code>	855, 859, 1009, 1022, 1030, 1035, 1055, 1059, 2658, 2689
<code>\l__enumext_topsep_vii_skip</code>	.. 1086, 1095, 1099
<code>\l__enumext_topsep_viii_skip</code>	.. 1108, 1130, 1134
<code>__enumext_use_key_ref:</code> 31, <u>407</u> , 407, 2235
<code>__enumext_use_key_ref_h:</code>	.. 31, <u>439</u> , 439, 2269
<code>\l__enumext_vspace_a_star_v_bool</code> 1251
<code>\l__enumext_vspace_a_star_vii_bool</code>	... 1273
<code>\l__enumext_vspace_a_star_viii_bool</code>	... 1284
<code>\l__enumext_vspace_a_star_X_bool</code> <u>70</u>
<code>__enumext_vspace_above:</code>	.. 47, <u>1219</u> , 1219, 2381
<code>__enumext_vspace_above_v:</code>	. 48, <u>1247</u> , 1247, 2529
<code>\l__enumext_vspace_above_v_skip</code>	.. 1249, 1253, 1255
<code>__enumext_vspace_above_vii:</code>	.. 48, <u>1269</u> , 1269, 2912
<code>\l__enumext_vspace_above_vii_skip</code>	1271, 1275, 1277
<code>__enumext_vspace_above_viii:</code>	. 48, <u>1269</u> , 1280
<code>\l__enumext_vspace_above_viii_skip</code>	1282, 1286, 1288
<code>\l__enumext_vspace_b_star_v_bool</code> 1262
<code>\l__enumext_vspace_b_star_vii_bool</code>	... 1295
<code>\l__enumext_vspace_b_star_viii_bool</code>	... 1306
<code>\l__enumext_vspace_b_star_X_bool</code> <u>70</u>
<code>__enumext_vspace_below:</code>	.. 48, <u>1233</u> , 1233, 2464
<code>__enumext_vspace_below_v:</code>	. 48, <u>1258</u> , 1258, 2597
<code>\l__enumext_vspace_below_v_skip</code>	.. 1260, 1264, 1266
<code>__enumext_vspace_below_vii:</code>	.. 49, <u>1291</u> , 1291, 2921
<code>\l__enumext_vspace_below_vii_skip</code>	1293, 1297, 1299
<code>__enumext_vspace_below_viii:</code>	. 49, <u>1291</u> , 1302
<code>\l__enumext_vspace_below_viii_skip</code>	1304, 1308, 1310
<code>__enumext_widest_from:nnn</code>	.. 33, <u>522</u> , 522, 537, 549
<code>\g__enumext_widest_label_tl</code>	22, 29, <u>40</u> , 311, 315, 319
<code>\l__enumext_wrap_label_opt_v_bool</code> 2096
<code>\l__enumext_wrap_label_opt_vii_bool</code>	85, 2984
<code>\l__enumext_wrap_label_opt_X_bool</code> <u>70</u>
<code>\l__enumext_wrap_label_v_bool</code>	2092, 2096, 2104, 2159
<code>\l__enumext_wrap_label_vii_bool</code>	85, 2983, 2988, 2996, 3074
<code>\l__enumext_wrap_label_X_bool</code> <u>70</u>
<code>__enumext_wrapper_label_v:n</code> 2161, 2637
<code>__enumext_wrapper_label_vii:n</code> 3077
<code>__enumext_zero_parsep:</code> 42, 906, <u>961</u> , 961
<code>enumext*</code> 4, <u>2830</u>
<code>enumXi</code> <u>283</u>
<code>enumXii</code> <u>283</u>
<code>enumXiii</code> <u>283</u>
<code>enumXiv</code> <u>283</u>
<code>enumXv</code> <u>283</u>
<code>enumXvi</code> <u>283</u>
<code>enumXvii</code> <u>283</u>
<code>enumXviii</code> <u>283</u>
Environments provide by enumext :	
<code>enumext*</code>	21–23, 25, 26, 28, 30–32, 35, 36, 38, 45, 48–51, 53–58, 63, 71, 82–84, 86–88, 91, 93
<code>enumext</code>	21–23, 25, 26, 28, 29, 31, 32, 34–44, 46–51, 53–58, 63–68, 70, 71, 73, 74, 78, 79, 81, 84, 88, 91, 92
<code>keyans*</code>	21–23, 25, 26, 28, 30–32, 35, 36, 38, 45, 48–50, 53, 54, 63, 91, 93
<code>keyanspic</code>	21–24, 28, 29, 32, 46, 49, 50, 54, 60–62, 76–79, 92
<code>keyans</code>	21–24, 26, 28, 29, 32, 34–38, 40, 43, 44, 46–50, 53, 54, 60–62, 66–68, 74, 76–78, 81, 91, 92
Environments:	
<code>enumext*</code> 69
<code>keyans*</code> 69
<code>list</code> 25, 26, 67, 68, 70
<code>lrbox</code> 79, 86, 87
<code>minipage</code> 25–27, 38, 41, 76, 78, 79, 86, 87
<code>multicols</code> 39–41, 46, 72, 73, 75, 76
exp commands:	
<code>\exp_after:wN</code> 3154
<code>\exp_args:Ne</code> 2320, 3142
<code>\exp_not:N</code>	149, 314, 403, 424, 434, 635, 649, 650, 661, 662, 673, 674, 1738, 1844, 1845, 1926, 1966, 1967, 3151
<code>\exp_not:n</code>	403, 404, 424, 425, 434, 435, 636, 1527, 1534, 1703, 1714, 1724, 1738, 1739, 1816, 1898, 1928, 1930, 2338, 2351, 2892, 2905
F	
<code>\fbox</code> 1477
file commands:	
<code>\file_input_stop:</code> 3451
<code>first</code> <u>687</u>
<code>font</code> <u>323</u>

<code>\footnote</code>	63
<code>\footnote</code>	63, 2009
<code>\footnotemark</code>	2019
<code>\footnotesize</code>	1845, 1967
<code>\footnotetext</code>	2003
G	
<code>\getkeyans</code>	13, 88, <u>3140</u>
group commands:	
<code>\group_begin:</code>	1665, 1843, 1965, 3053, 3072, 3162, 3196
<code>\group_end:</code>	1674, 1850, 1973, 3082, 3094, 3164, 3203
H	
<code>\hbadness</code>	3101
hbox commands:	
<code>\hbox_set:Nn</code>	303
<code>\hfill</code>	353, 357, 362, 363, 1173, 1191, 1738, 1926, 2807
hook commands:	
<code>\hook_gput_code:nnn</code>	9, 179, 183, 221
<code>\hook_gset_rule:nnnn</code>	222
<code>\hspace</code>	3112
<code>\hyperlink</code>	58, 61
<code>\hyperlink</code>	1738, 1926
<code>\hypertarget</code>	27
<code>\hypertarget</code>	251
I	
<code>\IfHyperBoolean</code>	229
<code>\IfPackageLoadedTF</code>	11, 225, 239
<code>\ignorespaces</code>	638
int commands:	
<code>\int_add:Nn</code>	2767
<code>\int_case:nn</code>	812, 963
<code>\int_compare:nNnTF</code>	417, 441, 888, 1007, 1152, 1156, 1160, 1430, 1456, 1657, 1661, 1858, 1881, 1909, 2306, 2360, 2365, 2404, 2420, 2434, 2453, 2478, 2514, 2518, 2547, 2572, 2585, 2605, 2609, 2664, 2737, 2747, 2763, 2856, 2941, 2951, 3107, 3116, 3132, 3262
<code>\int_compare_p:nNn</code>	1396, 1759, 1769, 1781, 1782, 1797, 1799, 2312, 2864
<code>\int_decr:N</code>	2766
<code>\int_eval:n</code>	1550, 1811, 1845, 1893, 1967, 2217, 2220, 2264, 2267, 2755
<code>\int_from_alph:n</code>	516, 530
<code>\int_from_roman:n</code>	518, 532
<code>\int_gadd:Nn</code>	1417, 2768, 3043
<code>\int_gincr:N</code>	1420, 1670, 1937, 2040, 2041, 2071, 2072, 2394, 2538, 2961, 3047, 3048
<code>\int_gset:Nn</code>	1346, 1357, 2017
<code>\int_gset_eq:NN</code>	1399, 1402, 1404, 1405, 1406, 1407, 1408, 1409, 2014, 2466, 2469, 2922, 2925
<code>\int_gzero:N</code>	1181, 1198, 1440, 1441, 1442, 1443, 1444, 1466, 1467, 1468, 1469, 1470, 2458, 2590, 3125
<code>\int_if_exist:NTF</code>	1333, 1381, 1383, 1385, 1387, 1389, 1391, 2467, 2923
<code>\int_incr:N</code>	2305, 2510, 2663, 2855, 2960
<code>\int_mod:nn</code>	3118
<code>\int_new:N</code>	20, 21, 22, 23, 24, 36, 38, 58, 74, 86, 91, 99, 112, 113, 119, 120, 121, 124, 125, 138, 155, 156, 157, 158, 159, 1335, 1382, 1384, 1386, 1388, 1390, 1392
<code>\int_set:Nn</code>	512, 516, 518, 1425, 1448, 1524, 1698, 2702, 2703, 2725, 2736, 2742, 2758, 3101, 3258
<code>\int_set_eq:NN</code>	2333, 2765, 2887
<code>\int_step_function:nnN</code>	1775, 1789, 1804
<code>\int_step_inline:nnn</code>	2704, 3285
<code>\int_to_roman:n</code>	187, 1755, 1793

<code>\int_use:N</code>	889, 1348, 1359, 1418, 2220, 2239, 2267, 2321, 2405, 2414, 2429, 2435, 2740, 2741, 2753, 3044
<code>\int_zero:N</code>	3110
<code>\c_one_int</code>	2725, 2744, 2750, 2756, 2760, 2763
<code>\c_zero_int</code>	1396, 1759, 1769, 1781, 1782, 1797, 1799, 2312, 2864, 2941, 2951, 3121
<code>\item</code>	26, 37, 38, 55, 64, 76, 77, 79, 82
<code>\item</code>	64, 65, 84, 86, 214, 1569, 1575, 1600, 1608, 1695, 1911, 1914, 2079, 2113, 2842, 2844
<code>\item*</code>	5, 11, <u>2111</u>
<code>item-pos*</code>	<u>1985</u>
<code>item-sym*</code>	<u>1985</u>
<code>\itemindent</code>	22, 68
<code>\itemindent</code>	67
<code>itemindent</code>	<u>602</u>
<code>\itemsep</code>	78, 79
<code>\itemsep</code>	2678, 2684
<code>\itemwidth</code>	2732, 2776, 2780

K

keyans	11, <u>2486</u>
keyans*	11
keyanspic	12, <u>2639</u>

Keys for environments provide by [enumext](#):

above*	23, 47, 48
above	23, 47, 48, 71, 75, 83
after	36–38, 73, 76, 83
align	23, 30, 66, 86
before*	36, 37, 71, 83
before	36–38, 75
below*	23, 47–49
below	23, 47–49, 73, 76, 83
check-ans	23–25, 49, 51, 56, 62, 64, 65, 71, 73, 87, 92
columns-sep*	23, 53, 70, 83
columns-sep	38, 54, 70, 72, 75, 83
columns*	23, 53, 70, 83
columns	22, 38, 41, 47, 54, 70, 72, 75, 83
first	36–38, 86
font	29, 66, 86
item-pos*	56, 57, 63
item-sym*	22, 56, 57, 63, 64
item*-sep	64
itemindent	23, 35, 66, 86
itemsep	34, 69
labelsep	29, 64, 68, 86
labelwidth	28, 29, 32, 33, 68
label	22, 28, 29, 32, 33, 79
lisparindent	69
list-indent	22, 35, 78
list-offset	35
listparindent	35, 86
mark-ans	24, 53, 59
mark-pos	53
mark-ref	24, 53, 58, 60
mini-env	22, 38, 41, 46, 47, 63, 71, 75, 81, 83
mini-sep	22, 38, 71, 75
miniright*	22, 38
miniright	22, 38, 45, 81
minirigth*	25
minirigth	25
no-store	24, 50, 51
noitemsep	34, 42
nosep	34, 42
parindent	69
parsep	34, 69, 86

partopsep	34
ref	25, 30, 31
resume	22, 49, 50, 68, 73, 83
rightmargin	35
save-ans	23, 49, 50, 54, 56, 60, 61, 65, 73, 74, 77, 83, 88
save-key	24
show-ans	24, 53, 56, 57, 59, 65
show-length	26, 36, 68, 91
show-pos	24, 53, 56, 57, 59, 65
start	23, 26, 33, 34, 68
store-brk	56, 57
store-ref	24, 27, 53, 57, 58, 60, 61, 65
topsep	34
widest	22, 26, 33, 34
wrap-ans	53, 56, 59
wrap-label*	29, 64, 66, 85, 86
wrap-label	29, 66, 85, 86
keys commands:	
\keys_define:nn	325, 347, 376, 457, 477, 493, 540, 560, 604, 623, 680, 689, 768, 785, 1204, 1313, 1322, 1366, 1474, 1503, 1521, 1641, 1987, 3166, 3228
\l_keys_key_str	3319
\keys_set:nn	339, 792, 1209, 1214, 1687, 2320, 2525, 2874, 3230, 3231, 3232, 3233, 3234, 3235, 3236, 3237, 3238, 3239, 3240, 3241, 3279
L	
l internal commands:	
\l_enumext_store_internal_ref:	57
label	455, 475, 493
Labels provide by enumext:	
\Alph*	28, 29
\Roman*	28, 29
\alph*	28, 29
\arabic*	28-30
\roman*	28, 29
\labelsep	79
\labelsep	2679, 2682
labelsep	323
\labelwidth	29, 79
\labelwidth	2679, 2680
labelwidth	323
\leftmargin	22, 68
\leftmargin	67, 2679
legacy commands:	
\legacy_if:nTF	3027, 3030
\legacy_if_gset_false:n	882
\legacy_if_set_false:n	3029
\legacy_if_set_true:n	2989, 3014, 3021, 3034
\linewidth	71, 75
\linewidth	2389, 2535, 2701, 2728, 2789
\list	26
\list	212
list-indent	602
list-offset	602
\listparindent	2681
listparindent	602
\lrbox	3054
M	
\makebox	79
\makebox	1631, 1633, 2133, 3068, 3076, 3080
\makelabel	64, 66, 67, 79
\makelabel	66, 67, 2139, 2155
\makesavenoteenv	245
mark-ans	1472

mark-pos	1472, 1501
mark-ref	1472
mini-env	766
mini-sep	766
\minipage	27
\minipage	218
\miniright	9, 46, 1150, 2456, 2588
\miniright*	9
mode commands:	
\mode_if_vertical:TF	837, 865, 988, 1067
\mode_leave_vertical:	635, 649, 661, 673, 1600, 1608, 1629, 2131, 3066
msg commands:	
\msg_error:nn	2516, 2520, 2607, 2666, 2858, 3242
\msg_error:nnn	1154, 1158, 1183, 1200, 3156, 3161, 3225, 3295
\msg_error:nnnn	1655, 1659, 1663, 2508, 2603, 2611
\msg_fatal:nn	2307
\msg_fatal:nnn	277
\msg_info:nnn	13, 16, 227, 241
\msg_line_context:	3323, 3328, 3333, 3348, 3371, 3375, 3379, 3384, 3389, 3394, 3399, 3403, 3408, 3413, 3417, 3422, 3426, 3431, 3436, 3441, 3445, 3449
\msg_new:nnn	3296, 3300, 3304, 3308, 3313, 3317, 3321, 3326, 3331, 3346, 3361, 3368, 3373, 3377, 3382, 3387, 3392, 3397, 3401, 3406, 3411, 3415, 3420, 3424, 3429, 3434, 3439, 3443, 3447
\msg_term:nnn	1433, 1459
\msg_term:nnnn	2229, 2239, 2274, 2279
\msg_warning:nn	2455, 2587
\msg_warning:nnn	1437, 1463
\msg_warning:nnnn	1944, 2171, 2176, 2739, 2752
\multicolsep	72, 75
\multicolsep	2419, 2560
N	
\NeedsTeXFormat	3
\newcounter	280
\NewDocumentCommand	1150, 1651, 2599, 3140, 3194, 3249
\NewDocumentEnvironment	2284, 2486, 2639, 2830
\newlabel	28
\newlabel	263
no-store	1364
\noindent	82
\noindent	2396, 2540, 2798, 2843, 3109
\nointerlineskip	2396, 2540, 2798
noitemsep	558
\nopagebreak	848, 876, 999, 1078, 1141, 1147
\normalfont	1844, 1966
nosep	558
P	
Packages:	
enumext	21, 49, 67, 77, 91
enumitem	28
expl3	79
footnotehyper	27
hyperref	24, 25, 27, 28, 31, 58, 61, 86, 91
lua-visual-debug	41
multicol	21, 91
shortlst	79
\par	848, 876, 999, 1078, 1141, 1147, 1176, 1193, 1823, 2440, 2460, 2577, 2592, 2713, 2816, 2823, 3109, 3123
\parindent	3086
\parsep	39, 42, 78, 79

`\parsep` 1601, 1609, 2259, 2678, 2685, 2690
`parsep` 558
`\parskip` 3087
`\partopsep` 79
`\partopsep` 2260, 2683
`partopsep` 558
peek commands:
 `\peek_meaning:NTF` 2966, 2980, 2997, 3008
 `\peek_meaning_remove:NTF` 2973
 `\peek_remove_spaces:n` 2117
`\phantomsection` 27
`\phantomsection` 252
prg commands:
 `\prg_do_nothing:` 256
 `\prg_new_protected_conditional:Npnn` ... 198
 `\prg_replicate:nn` 207, 3366
 `\prg_return_false:` 202
 `\prg_return_true:` 201
`\printkeyans` 13, 88, 3194
prop commands:
 `\prop_count:N` 1550, 1811, 1847, 1893, 1969
 `\prop_gput:Nnn` 1548
 `\prop_if_exist:NTF` 1546, 3160
 `\prop_item:Nn` 3163
 `\prop_new:N` 1547
`\ProvidesExplPackage` 4

R

`\raggedcolumns` 2428, 2566
`\ref` 58, 60
`ref` 455, 475
`\refstepcounter` 3036
regex commands:
 `\regex_match:nnTF` 200, 515, 517, 529, 531, 2331, 2344, 2885, 2898
 `\regex_replace_once:nnN` 391
`\renewcommand` 403, 424, 434
`\RenewDocumentCommand` ... 2009, 2079, 2113, 2139, 2155
`\RequirePackage` 17
`resume` 1313
`resume*` 1313
`rightmargin` 602
`\Roman` 29, 33
`\Roman` 299
`\roman` 29, 33
`\roman` 300, 473, 3182

S

`save-ans` 1313
scan commands:
 `\scan_stop:` 79, 2692, 2842, 3151, 3154
seq commands:
 `\seq_clear:N` 3256
 `\seq_const_from_clist:Nn` 3244
 `\seq_count:N` 2652, 3260
 `\seq_gclear:N` 2007, 2008
 `\seq_gput_right:Nn` 1558, 2020, 2021
 `\seq_if_empty:NTF` 2026, 3209, 3274
 `\seq_if_exist:NTF` 1556, 3207
 `\seq_item:Nn` 2710
 `\seq_map_function:NN` 3265
 `\seq_map_inline:Nn` .. 3214, 3219, 3253, 3275, 3276
 `\seq_map_pairwise_function:NNN` 2028
 `\seq_new:N` 92, 93, 110, 139, 140, 1557
 `\seq_pop_left:NN` 3264

`\seq_put_right:Nn` 2613, 3272, 3289
`\seq_set_from_clist:Nn` 3257
`\seq_set_map_e:NNn` 3266
`\setcounter` .. 526, 530, 532, 2217, 2219, 2264, 2266, 2657
`\setenumext` .. 5–8, 89, 3170, 3175, 3180, 3185, 3190, 3249
`\setlength` 1602, 1610
`show-ans` 1472, 1501
`show-length` 678
skip commands:
 `\skip_add:Nn` . 817, 823, 829, 839, 843, 867, 871, 968, 974, 980, 990, 994, 1016, 1069, 1073, 2678
 `\skip_eval:n` 1601, 1609
 `\skip_gset:Nn` 1089, 1093, 1097
 `\skip_gzero_new:N` 1084, 1085
 `\skip_horizontal:N` 650, 662, 674, 3069, 3083
 `\skip_horizontal:n` 636, 1630, 1638, 2132, 2134, 3067
 `\skip_if_eq:nnTF` . 815, 821, 827, 891, 925, 966, 972, 978, 1009, 1014, 1035, 1086, 1108, 1221, 1235, 1249, 1260, 1271, 1282, 1293, 1304
 `\skip_new:N` 54, 55, 59, 60, 61, 62, 63, 114, 169
 `\skip_set:Nn` . 800, 804, 853, 857, 894, 898, 902, 909, 913, 917, 928, 933, 937, 943, 948, 953, 1011, 1012, 1013, 1020, 1024, 1028, 1037, 1042, 1046, 1049, 1053, 1057, 1088, 1092, 1110, 1114, 1118, 1124, 1128, 1132, 2672, 2686
 `\skip_set_eq:NN` 2212, 2258, 2259, 3086, 3087
 `\skip_use:N` 802, 806, 841, 845, 849, 869, 873, 892, 911, 920, 926, 931, 935, 946, 950, 951, 956, 992, 996, 1022, 1222, 1226, 1229, 1236, 1240, 1243, 2440
 `\skip_zero:N` 2260, 2419, 2560, 2683, 2684
 `\skip_zero_new:N` 1004, 1005, 1006, 1083, 1105, 1106, 1107
 `\c_zero_skip` . 815, 821, 827, 892, 926, 966, 972, 978, 1009, 1014, 1035, 1086, 1108, 1222, 1236, 1249, 1260, 1271, 1282, 1293, 1304

`\small` 3172, 3177, 3182, 3187, 3192
`\star` 1991
`start` 538
`\stepcounter` 2013, 2620
`store-ref` 1472
str commands:
 `\c_backslash_str` 3375, 3384, 3385, 3389, 3390, 3394, 3395, 3426, 3427, 3431, 3436, 3437
 `\c_colon_str` 1810, 1892, 3151
 `\str_count:n` 207, 3366
 `\str_if_eq:nnTF` 2222, 2270
 `\str_if_eq_p:nn` 2215, 2263
 `\str_if_in:nnTF` 3147
 `\str_new:N` 109, 164
 `\str_set:Nn` ... 379, 380, 381, 1483, 1484, 1506, 1507
`\string` 245
`\strutbox` 896, 900, 904, 915, 919, 930, 939, 945, 955, 968, 974, 980, 1011, 1012, 1013, 1016, 1026, 1030, 1039, 1046, 1051, 1059, 1088, 1089, 1092, 1099, 1112, 1120, 1126, 1134, 2688

T

TeX and \TeX 2_ε commands:

`\@auxout` 261
`\protected@write` 261
text commands:
 `\text_expand:n` 3143
`\textasteriskcentered` 1480, 1494
`\thepage` 267

tl commands:

<code>\c_space_tl</code>	1867, 1918, 1959, 1982, 3333, 3348
<code>\tl_clear:N</code>	352, 358, 1684, 1857, 1908
<code>\tl_clear_new:N</code>	309
<code>\tl_const:Nn</code>	141, 293
<code>\tl_gclear:N</code>	...	1946, 2150, 2484, 2827, 3070, 3138
<code>\tl_gput_right:Nn</code>	294
<code>\tl_greplace_all:Nnn</code>	315
<code>\tl_gset:Nn</code>	1934, 2446, 2932, 3003
<code>\tl_gset_eq:NN</code>	311, 2059, 3063
<code>\tl_if_blank:nTF</code>	3061
<code>\tl_if_empty:nTF</code>	...	409, 443, 449, 1565, 1596, 1710, 1942, 2129, 3287
<code>\tl_if_novalue:nTF</code>	..	1685, 1696, 1865, 1916, 1958, 1981, 2011, 2036, 2055, 2060, 2090, 2650, 2872, 3251
<code>\tl_map_inline:Nn</code>	312, 389
<code>\tl_new:N</code>	32, 39, 41, 42, 75, 76, 77, 83, 84, 85, 87, 88, 89, 90, 96, 97, 107, 108, 118, 130, 131, 132, 135, 143, 144, 147, 148, 163, 166	
<code>\tl_put_left:Nn</code>	1573, 1606, 1693, 1952, 1975
<code>\tl_put_right:Nn</code>	310, 401, 422, 432, 1525, 1532, 1577, 1612, 1695, 1701, 1709, 1712, 1722, 1727, 1730, 1736, 1762, 1772, 1786, 1802, 1808, 1813, 1860, 1863, 1867, 1883, 1887, 1890, 1895, 1911, 1914, 1918, 1924, 1959, 1982, 2336, 2349, 2890, 2903, 3168, 3173, 3178, 3183, 3188	
<code>\tl_remove_all:Nn</code>	3286
<code>\tl_remove_once:Nn</code>	1750, 1877
<code>\tl_replace_all:Nnn</code>	314
<code>\tl_reverse:N</code>	1749, 1751, 1876, 1878
<code>\tl_set:Nn</code>	149, 279, 353, 357, 362, 363, 397, 416, 633, 647, 659, 671, 1331, 1479, 1493, 1841, 1963, 2057, 3284	
<code>\tl_set_eq:NN</code>	320, 398, 400, 419, 421, 429, 431, 1748,	

1875, 2102, 2106, 2631, 2633

<code>\tl_to_str:n</code>	3143
<code>\tl_trim_spaces:n</code>	310, 3272, 3284, 3290
<code>\tl_use:N</code>	...	316, 319, 411, 445, 451, 704, 708, 712, 716, 720, 724, 728, 732, 736, 740, 744, 748, 752, 756, 760, 764, 1635, 1755, 1763, 1774, 1788, 1793, 1805, 2044, 2050, 2075, 2093, 2097, 2105, 2133, 2141, 2142, 2149, 2157, 2158, 2164, 2291, 2492, 2636, 2821, 3073, 3084, 3088, 3197, 3198, 3199, 3200, 3201, 3268

token commands:

<code>\token_to_str:N</code>	263
<code>\topsep</code>	1602, 1610
<code>topsep</code>	<u>558</u>
<code>\typeout</code>	231, 234, 244, 245

U

<code>\u</code>	392
use commands:		
<code>\use:N</code>	208, 2146, 2293
<code>\use:n</code>	3149
<code>\use_none:nn</code>	255
<code>\usecounter</code>	2213, 2261

V

<code>\value</code>	2466, 2471, 2922, 2927
<code>\vspace</code>	883, 1226, 1229, 1240, 1243, 1253, 1255, 1264, 1266, 1275, 1277, 1286, 1288, 1297, 1299, 1308, 1310, 1601, 1609, 2647, 2658, 3124	

W

<code>widest</code>	<u>538</u>
<code>wrap-ans</code>	<u>1472</u>
<code>wrap-label</code>	<u>323</u>
<code>wrap-label*</code>	<u>323</u>