

enumext

ENUMERATE EXERCISE SHEETS

V1.0 2024-05-02^{*}

©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	102
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 2024-05-02.

[†]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. (a) $(x - 1)^2$

* ii. Yes, dnf

*

(b) $3(x + y + z)$

* iii. A. doesn’t exist

(c) i. False

* for now :(

ii. Very True!

* B. very good

(d) i. Yes

* C. obsolete
- Another very common thing to use in my work is “multiple choice questions”, for example:
1. First type of questions

(A) value

(C) value

(B) correct

(D) value

2. Second type of questions

I. $2\alpha + 2\delta = 90^\circ$

II. $\alpha = \delta$

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

(A) I only

(D) I and III only

(B) II only

(E) I, II, and III

(C) I and II only

★ 3. Third type of questions

(1) $2\alpha + 2\delta = 90^\circ$

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

(A) value


(D) value

(B) value


(E) value

(C) value


4. Question with image and label below:




(A)




(B)



(C)



(D)



(E)

5. Question with image on left side:


(A) value

(B) value

(C) value

(D) correct

(E) value



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

1. (a) (B) $x = 5$

* (e) (C) some note

*

(b)

(f) (B)

(c) (D)

(g) (D) “other note”

(d)

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”.

©2024 by Pablo González L

2 / 113

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.

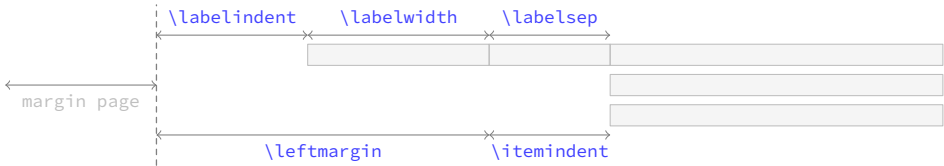


Figure 1: Representation of horizontal lengths in `enumitem`.

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.

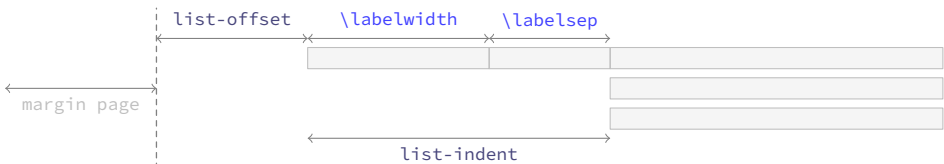


Figure 2: Representation of horizontal lengths concept in `enumext`.

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 multicol

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

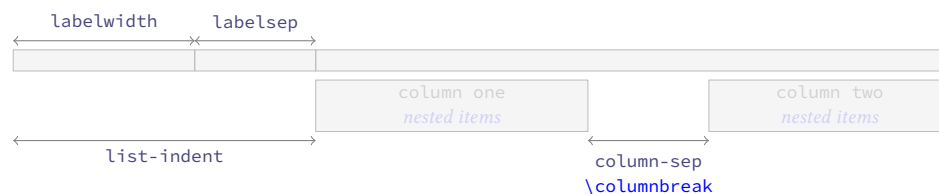


Figure 4: Representation of the two column output for a nested level in `enumext` environment.

The “non starred” version of the `multicols` environment is always used together with the `\raggedcolumns` command and is controlled by `columns` and `columns-sep` keys. The environment is available for all nesting levels, and can can together with the `mini-env` key. If you need to force a start a new column `\columnbreak` must be used (see §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 \TeX `\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. TeX will enter *vertical mode* and apply this value to the “top” and “bottom” the environment or nested level.

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

Set the *vertical space* between items, beyond the `parsep`. Internally sets the value of `\itemsep` for the current level. The default 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` `\miniright*` The `\miniright` command close the `minipage` environment on the “left side” and opens the `minipage` environment on the “right side” by starting it with the `\centering` command. It must be placed “after” the last `\item` of the current environment and “before” starting the material to be placed on the “right side”. The *starred 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

- ★ 1. Text containing our instructions or questions.

*

first answer

2. Text containing our instructions or questions.

(a) Question.

*

second answer
3. Text containing our instructions or questions.

*

third answer

4. Text containing our instructions or questions.

*

fourth answer

```
\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[\keys]{\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}[<number above, number below>]\anspic{<drawing>}\anspic*{<content>}{<drawing>}
```

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

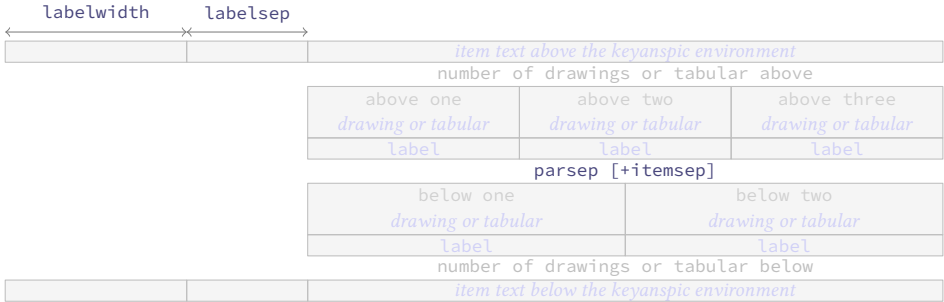


Figure 6: Representation of the `keyanspic` environment with optional argument `[3,2]` in `enumext`.

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

4.6.1 The command \anspic

```
\anspic \anspic{<drawing or tabular>}
\anspic*{<content>}{<drawing or tabular>}
```

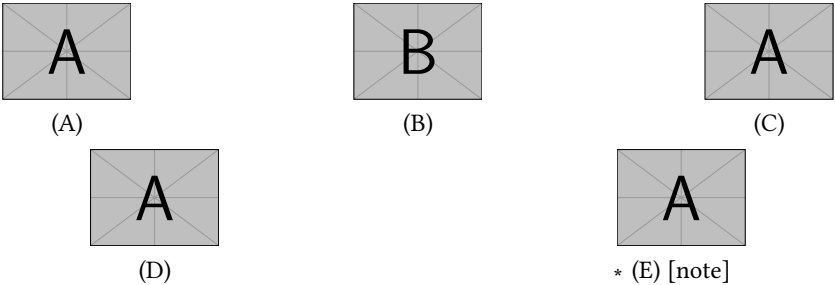
The `\anspic` command take three arguments, the *starred version* “*” store the current `<label>` next to the `<content>` (if it is present) in `<store name>` set by `save-ans` key.

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. (a) $3(x + y + z)$

(b) i. Very True!

(c) i. Yes

*

ii. A. very good

B. 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. (a) B

(b) A
- (c) B

(d) 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. (a) B

(b) A

(c) B

(d) A
- *


*

*

*

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

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

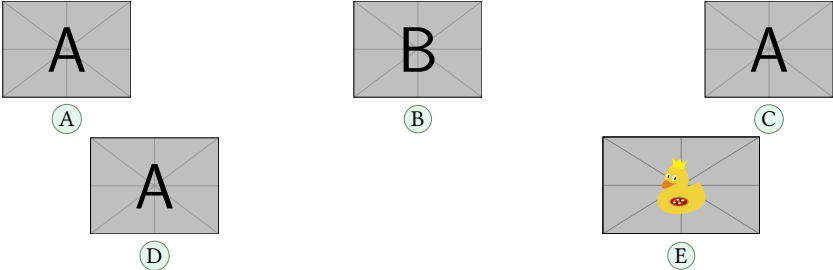
 value

D

 value

E

 value
4. Question with image and label below:



5. Question with image on left side:
- A

 value

B

 value

C

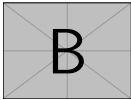
 value

D

 correct

E

 value



Test keys

1. (a) B $x = 5$

(b)

(c) D

(d)

*


 (e) C some note

(f) B

*

 (g) D other note

Example 4

A “simple worksheet” using ducks :) .

- ca. 1.

Factor $x^2 - 2x + 1$

ca. 2.

Factor $3x + 3y + 3z$

The following questions need to be cuaqtified :)

ca. 3.

True False

(a)

 $\alpha > \delta$

(b)

 \LaTeX is cool?

ca. 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. (a) $(x - 1)^2$

(b) $3(x + y + z)$

(c) i. False

ii. Very True!

(d) i. Yes

*

 ii. Yes, dnf

*

 iii. A. doesn't exist for now :(

*

 B. very good

*

 C. obsolete

*

Example 5

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

1

Which choice best describes what happens in the passage?

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

2

Which choice best describes what happens in the passage?

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

3

Which choice best describes what happens in the passage?

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

4

Which choice best describes what happens in the passage?

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

1. (a) A) (c) B)
 (b) C) (d) 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 \LaTeX for the four levels `\textbullet`, `\textendash`, `\textasteriskcentered` and `\textperiodcentered` together with the `nosep` key to reduce the vertical spaces in the left side example and set the `label` key in *mathematical mode* for the right side as `\ast`, `\diamond`, `\circ` and `\star` for the four levels together with the `nosep` key

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

Fake description environment

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

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

Something A short *one-line* description text.

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

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

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

Something A short *one-line* description text.

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

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

Description indented by label

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

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

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

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

Something A short one-line description.

Something long A much longer description. Lorem ipsum dolor sit amet, 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 2024-05-02 – 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 / 113

R	\rightmargin	8
\raggedcolumns		4
\ref		4
	T	
	\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>.

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

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 (\LaTeX 3 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 \LaTeX to work correctly.

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

Now declare the `enumext` package.

```
4 \ProvidesExplPackage
5   {enumext}
6   {2024-05-02}
7   {1.0}
8   {Enumerate exercise sheets}
```

Finally 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. The boolean variables `\g__enumext_starred_bool` and `\g__enumext_standar_bool` will be set to “true” when the `enumext` and `enumext*` environments are not nested with each other.

```
20 \int_new:N \l__enumext_level_int
21 \int_new:N \l__enumext_level_h_int
22 \int_new:N \l__enumext_keyans_level_int
23 \int_new:N \l__enumext_keyans_level_h_int
24 \int_new:N \l__enumext_keyans_pic_level_int
25 \bool_new:N \l__enumext_starred_bool
26 \bool_new:N \g__enumext_starred_bool
```

```

27 \bool_new:N \l__enumext_standar_bool
28 \bool_new:N \g__enumext_standar_bool
29 \bool_new:N \l__enumext_keyans_env_bool

```

(End of definition for `\l__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 { \l__enumext_minipage_left_#1_dim }
67   \bool_new:c { \l__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.)

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

The integer variable `\l__enumext_start_X_int` are used by the `start` key (§10.12), the token list `\l__enumext_fake_item_indent_X_tl` is used by `itemindent` key, the variables `\l__enumext_label_fill_left_X_tl` and `\l__enumext_label_fill_right_X_tl` are used by the `align` key (§10.10). The boolean vars `\l__enumext_vspace_a_star_X_bool`, `\l__enumext_vspace_b_star_X_bool` are used by `above`, `above*`, `below` and `below*` keys

```
70 \cs_set_protected:Npn \__enumext_tmp:n #1
71 {
72   \bool_new:c { \l__enumext_wrap_label_#1_bool }
73   \bool_new:c { \l__enumext_wrap_label_opt_#1_bool }
74   \int_new:c { \l__enumext_start_#1_int }
75   \tl_new:c { \l__enumext_fake_item_indent_#1_tl }
76   \tl_new:c { \l__enumext_label_fill_left_#1_tl }
77   \tl_new:c { \l__enumext_label_fill_right_#1_tl }
78   \bool_new:c { \l__enumext_vspace_a_star_#1_bool }
79   \bool_new:c { \l__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 `\l__enumext_wrap_label_X_bool` and others.)

```
\l__enumext_store_active_bool
\l__enumext_store_name_tl
\g__enumext_store_name_tl
\l__enumext_store_anskey_arg_tl
\l__enumext_store_columns_join_int
\l__enumext_store_keyans_label_tl
\l__enumext_keyans_tmpa_tl
```

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

The variable `\l__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 `\l__enumext_store_anskey_arg_tl` stores the contents of `\anskey` (§10.24) and the variable `\l__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 `\l__enumext_keyans_tmpa_tl` is a temporary variable used by `keyans` and `keyanspic` at various points.

```
82 \bool_new:N \l__enumext_store_active_bool
83 \tl_new:N \l__enumext_store_name_tl
84 \tl_new:N \g__enumext_store_name_tl
85 \tl_new:N \l__enumext_store_anskey_arg_tl
86 \int_new:N \l__enumext_store_columns_join_int
87 \tl_new:N \l__enumext_store_keyans_label_tl
88 \tl_new:N \l__enumext_keyans_tmpa_tl
```

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

```
\l__enumext_setkey_tmpa_tl
\l__enumext_setkey_tmpp_tl
\l__enumext_setkey_tmpa_int
\l__enumext_setkey_tmpa_seq
\l__enumext_setkey_tmpp_seq
```

Internal variables used by the command `\setenumext` (§10.39).

```
89 \tl_new:N \l__enumext_setkey_tmpa_tl
90 \tl_new:N \l__enumext_setkey_tmpp_tl
91 \int_new:N \l__enumext_setkey_tmpa_int
92 \seq_new:N \l__enumext_setkey_tmpa_seq
93 \seq_new:N \l__enumext_setkey_tmpp_seq
```

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

```
\l__enumext_store_opt_X_tl
\l__enumext_print_keyans_X_tl
\l__enumext_store_columns_X_bool
\l__enumext_store_columns_X_int
\l__enumext_store_columns_sep_X_bool
\l__enumext_store_columns_sep_X_dim
\l__enumext_store_upper_level_X_bool
```

Internal variables used by [$\langle key = val \rangle$] in `enumext` and `enumext*` environment, the command `\printkeyans` (§10.38) and the keys `columns*` and `columns-sep*`.

```
94 \cs_set_protected:Npn \l__enumext_tmp:n #1
95 {
96   \tl_new:c { \l__enumext_store_opt_#1_tl }
97   \tl_new:c { \l__enumext_print_keyans_#1_tl }
98   \bool_new:c { \l__enumext_store_columns_#1_bool }
99   \int_new:c { \l__enumext_store_columns_#1_int }
100   \bool_new:c { \l__enumext_store_columns_sep_#1_bool }
101   \dim_new:c { \l__enumext_store_columns_sep_#1_dim }
102   \bool_new:c { \l__enumext_store_upper_level_#1_bool }
103 }
104 \clist_map_inline:nn { i, ii, iii, iv, vii } { \l__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_store_ans_bool
\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_with_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_store_ans_bool
116 \bool_new:N \l__enumext_check_ans_bool
117 \bool_new:N \g__enumext_check_ans_show_bool
118 \bool_new:N \g__enumext_check_ans_show_h_bool
119 \tl_new:N \g__enumext_check_ans_item_tl
120 \int_new:N \l__enumext_compare_items_ans_int
121 \int_new:N \g__enumext_count_item_with_ans_int
122 \int_new:N \g__enumext_count_item_all_int
123 \cs_set_protected:Npn \l__enumext_tmp:n #1
124 {
125   \int_new:c { \g__enumext_count_level_#1_int }
126   \int_new:c { \g__enumext_count_item_#1_int }
127 }
128 \clist_map_inline:nn { i, ii, iii, iv, vii } { \l__enumext_tmp:n {#1} }
```

(End of definition for `\l__enumext_store_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`.

```
129 \bool_new:N \l__enumext_hyperref_bool
130 \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.

```

131 \tl_new:N \l__enumext_newlabel_arg_one_tl
132 \tl_new:N \l__enumext_newlabel_arg_two_tl
133 \tl_new:N \l__enumext_store_write_aux_file_tl
134 \cs_set_protected:Npn \__enumext_tmp:n #1
135 {
136   \tl_new:c { l__enumext_label_copy_#1_tl }
137 }
138 \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`.

```

139 \int_new:N \g__enumext_footnote_int
140 \seq_new:N \g__enumext_footnote_arg_seq
141 \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`.)

```

\c__enumext_counter_style_tl
\l__enumext_ref_key_arg_tl
\l__enumext_ref_aux_tl
\l__enumext_the_counter_X_tl
\l__enumext_counter_style_for_ref_X_tl

```

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

```

142 \tl_const:Nn \c__enumext_counter_style_tl
143   { { arabic } { roman } { Roman } { alph } { Alph } }
144 \tl_new:N \l__enumext_ref_key_arg_tl
145 \tl_new:N \l__enumext_ref_aux_tl
146 \cs_set_protected:Npn \__enumext_tmp:n #1
147 {
148   \tl_new:c { l__enumext_counter_style_for_ref_#1_tl }
149   \tl_new:c { l__enumext_the_counter_#1_tl }
150   \tl_set:ce { l__enumext_the_counter_#1_tl } { \exp_not:c { theenumX#1 } }
151 }
152 \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.)

```

\l__enumext_item_starred_X_bool
\l__enumext_item_column_pos_X_int
\g__enumext_item_count_all_X_int
\l__enumext_joined_item_X_int
\l__enumext_joined_item_aux_X_int
\l__enumext_tmpa_X_int
\l__enumext_item_text_X_box
\l__enumext_joined_width_X_dim
\l__enumext_item_width_X_dim
\g__enumext_item_symbol_aux_X_tl
\l__enumext_align_label_X_str
\g__enumext_minipage_active_X_bool
\g__enumext_miniright_code_X_tl
\g__enumext_minipage_center_X_bool
\g__enumext_minipage_right_X_dim
\g__enumext_minipage_right_X_skip

```

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

```

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

```

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

```
\c__enumext_all_envs_clist
```

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

```

173 \clist_const:Nn \c__enumext_all_envs_clist
174 {
175   {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv},
176   {keyans}{v}, {enumext*}{vii}, {keyans*}{viii}
177 }

```

(End of definition for `\c__enumext_all_envs_clist`.)

10.5 Some utility functions

`__enumext_at_begin_document:n`

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

```
178 \cs_new_protected:Npn \__enumext_at_begin_document:n #1
179 {
180   \hook_gput_code:nnn {begindocument} {enumext} { #1 }
181 }
```

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

`__enumext_after_env:nn`

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.

```
182 \cs_new_protected:Npn \__enumext_after_env:nn #1 #2
183 {
184   \hook_gput_code:nnn {env/#1/after} {enumext} {#2}
185 }
```

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

`__enumext_level:`

Function for check current level in `enumext`.

```
186 \cs_new:Nn \__enumext_level:
187 {
188   \int_to_roman:n { \__enumext_level_int }
189 }
```

(End of definition for `__enumext_level:.`)

`__enumext_level_set:n`

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

`__enumext_level_end:n`

```
190 \cs_new:Npn \__enumext_level_set:n #1
191 {
192   \cs_set_eq:cN { \__enumext_level_#1: } \__enumext_level:
193   \cs_set:Nn \__enumext_level: { #1 }
194 }
195 \cs_new:Npn \__enumext_level_end:n #1
196 {
197   \cs_set_eq:Nc \__enumext_level: { __enumext_level_#1: }
198 }
```

(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` keys. This function is taken directly from the answer given by Henri Menke in [How to test if an expl3 function argument is an integer expression?](#).

`__enumext_if_is_int:nF`

`__enumext_if_is_int:nTF`

```
199 \prg_new_protected_conditional:Npnn \__enumext_if_is_int:n #1 { T, F, TF }
200 {
201   \regex_match:nnTF { ^[\+|-]?[\d]+$ } {#1} % $
202   { \prg_return_true: }
203   { \prg_return_false: }
204 }
```

(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.

```
205 \cs_new:Npn \__enumext_show_length:nnn #1 #2 #3
206 {
207   * ~ #2
208   \prg_replicate:nn { 14 - \str_count:n {#2} } { ~ }
209   = ~ \use:c { #1_use:c } { \__enumext_#2_#3_#1 } \\
210 }
```

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

`__enumext_zero_count_level:`

Internal function used by `check-ans` key.

```
211 \cs_set_protected:Nn \__enumext_zero_count_level:
212 {
213   \cs_set_protected:Npn \__enumext_tmp:n ##1
214   {
215     \int_gzero:c { g__enumext_count_level_##1_int }
216   }
217   \clist_map_inline:nn { i, ii, iii, iv, vii } { \__enumext_tmp:n {##1} }
218 }
```

(End of definition for `__enumext_zero_count_level:.`)

10.6 Copying list and minipage environments

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

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

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

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

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

```
219 \__enumext_at_begin_document:n
220 {
221   \cs_new_eq:NN \__enumext_start_list:nn \list
222   \cs_new_eq:NN \__enumext_stop_list: \endlist
223   \cs_new_eq:NN \__enumext_item_std:w \item
224 }
```

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

The `minipage` environment provided by \LaTeX has the following (simplified) plain form:

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

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

```
\__enumext_minipage:w
\__enumext_endminipage:
```

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

```
225 \__enumext_at_begin_document:n
226 {
227   \cs_new_eq:NN \__enumext_minipage:w \minipage
228   \cs_new_eq:NN \__enumext_endminipage: \endminipage
229 }
```

(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.

```
230 \hook_gput_code:nnn { begindocument } { enumext } { \__enumext_after_hyperref: }
231 \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”.

```
232 \cs_new_protected:Nn \__enumext_after_hyperref:
233 {
234   \IfPackageLoadedTF { hyperref }
235   {
236     \msg_info:nnn { enumext } { package-load } { hyperref }
237     \bool_set_true:N \l__enumext_hyperref_bool
238     \IfHyperBoolean{hyperfootnotes}
239     {
240       \typeout{hyperfootnotes=true}
241       \bool_set_true:N \l__enumext_footnotes_key_bool
242     }
243     { \typeout{hyperfootnotes=false} }
244   }
245   { }
```

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`.

```
246   \bool_if:NT \l__enumext_footnotes_key_bool
247   {
248     \IfPackageLoadedTF { footnotehyper }
249     {
```

```

250         \msg_info:nnn { enumext } { package-load } { footnotehyper }
251     }
252     {
253         \typeout{No ~ footnotehyper ~ load}
254         \typeout{Load ~ and ~ use ~ \string\makesavenoteenv{enumext*}}
255         \bool_set_false:N \l__enumext_footnotes_key_bool
256     }
257 }

```

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.

```

258     \bool_if:NTF \l__enumext_hyperref_bool
259     {
260         \cs_new_eq:NN \__enumext_hypertarget:nn \hypertarget
261         \cs_new_eq:NN \__enumext_phantomsection: \phantomsection
262     }
263     {
264         \cs_new_eq:NN \__enumext_hypertarget:nn \use_none:nn
265         \cs_new_eq:NN \__enumext_phantomsection: \prg_do_nothing:
266     }
267 }

```

(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.

```

268 \cs_new_protected:Npn \__enumext_newlabel:nn #1 #2
269 {
270     \protected@write \@auxout { }
271     {
272         \token_to_str:N \newlabel {#1}
273         {
274             {#2}
275             \bool_if:NT \l__enumext_hyperref_bool
276             { { \thepage } {#2} {#1} }
277             { }
278         }
279     }
280     \__enumext_hypertarget:nn {#1} { }
281     \__enumext_phantomsection:
282 }

```

(End of definition for `__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.

```

283 \cs_new_protected:Npn \__enumext_define_counters:Nn #1 #2
284 {
285     \cs_if_exist:cTF { c@ #2 }
286     { \msg_fatal:nnn { enumext } { counters } { #2 } }
287     {
288         \tl_set:Nn #1 { #2 }
289         \newcounter { #2 }
290     }
291 }

```

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

enumXi The counters created here are enumXi, enumXii, enumXiii and enumXiv for enumext environment,
enumXii enumXv for keyans environment, enumXvi for keyanspic environment, enumXvii for enumext* and
enumXiii enumXviii for the keyans* environments.

```
enumXiv 292 \__enumext_define_counters:Nn \__enumext_counter_i_tl { enumXi }
enumXv 293 \__enumext_define_counters:Nn \__enumext_counter_ii_tl { enumXii }
enumXvi 294 \__enumext_define_counters:Nn \__enumext_counter_iii_tl { enumXiii }
enumXvii 295 \__enumext_define_counters:Nn \__enumext_counter_iv_tl { enumXiv }
enumXviii 296 \__enumext_define_counters:Nn \__enumext_counter_v_tl { enumXv }
297 \__enumext_define_counters:Nn \__enumext_counter_vi_tl { enumXvi }
298 \__enumext_define_counters:Nn \__enumext_counter_vii_tl { enumXvii }
299 \__enumext_define_counters:Nn \__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.

```
300 \cs_new_protected:Npn \__enumext_register_counter_style:Nn #1 #2
301 {
302   \tl_const:cn { c__enumext_widest_ \cs_to_str:N #1 _tl } {#2}
303   \tl_gput_right:Nn \g__enumext_counter_styles_tl {#1}
304 }
305 \__enumext_register_counter_style:Nn \arabic { 0 }
306 \__enumext_register_counter_style:Nn \Alph { M }
307 \__enumext_register_counter_style:Nn \alph { m }
308 \__enumext_register_counter_style:Nn \Roman { VIII }
309 \__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.

```
310 \cs_new_protected:Npn \__enumext_label_width_by_box:Nn #1 #2
311 {
312   \hbox_set:Nn \l__enumext_label_width_by_box {#2}
313   \dim_set:Nn #1 { \box_wd:N \l__enumext_label_width_by_box }
314 }
315 \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.

```
316 \cs_new_protected:Npn \__enumext_label_style:Nnn #1 #2 #3
317 {
318   \tl_clear_new:N #1
319   \tl_put_right:Ne #1 { \tl_trim_spaces:n {#3} }
320   \tl_gset_eq:NN \g__enumext_widest_label_tl #1
321   \tl_map_inline:Nn \g__enumext_counter_styles_tl
322   {
323     \tl_replace_all:Nne #1 { ##1* } { \exp_not:N ##1 {#2} }
324     \tl_greplace_all:Nne \g__enumext_widest_label_tl { ##1* }
325     { \tl_use:c { c__enumext_widest_ \cs_to_str:N ##1 _tl } }
326   }
327   \__enumext_label_width_by_box:Nn \l__enumext_current_widest_dim
328   { \tl_use:N \g__enumext_widest_label_tl }
329   \tl_set_eq:cN { the #2 } #1
330 }
331 \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 Definition of keys `font`, `labelsep`, `labelwidth`, `wrap-label` and `wrap-label*` keys for `enumext` and `keyans` environments.

```

332 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
333 {
334   \keys_define:nn { enumext / #1 }
335   {
336     font      .tl_set:c   = { l__enumext_label_font_style_#2_tl },
337     font      .value_required:n = true,
338     labelsep  .dim_set:c   = { l__enumext_labelsep_#2_dim },
339     labelsep  .initial:n   = {0.3333em},
340     labelsep  .value_required:n = true,
341     labelwidth .dim_set:c   = { l__enumext_labelwidth_#2_dim },
342     labelwidth .value_required:n = true,
343     wrap-label .cs_set_protected:cp = { __enumext_wrapper_label_#2:n } ##1,
344     wrap-label .initial:n   = {##1},
345     wrap-label .value_required:n = true,
346     wrap-label* .code:n = {
347       \bool_set_true:c { l__enumext_wrap_label_opt_#2_bool }
348       \keys_set:nn { enumext / #1 } { wrap-label = {##1} }
349     },
350     wrap-label* .value_required:n = true,
351   }
352 }
353 \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.

```

354 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
355 {
356   \keys_define:nn { enumext / #1 }
357   {
358     align .choice:,
359     align / left .code:n =
360       {
361         \tl_clear:c { l__enumext_label_fill_left_#2_tl }
362         \tl_set:cn { l__enumext_label_fill_right_#2_tl } { \hfill }
363       },
364     align / right .code:n =
365       {
366         \tl_set:cn { l__enumext_label_fill_left_#2_tl } { \hfill }
367         \tl_clear:c { l__enumext_label_fill_right_#2_tl }
368       },
369     align / center .code:n =
370       {
371         \tl_set:cn { l__enumext_label_fill_left_#2_tl } { \hfill }
372         \tl_set:cn { l__enumext_label_fill_right_#2_tl } { \hfill }
373       },
374     align .initial:n = left,
375     align .value_required:n = true,
376   }
377 }
378 \clist_map_inline:nn
379 {
380   {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {keyans}{v}
381 }
382 { \__enumext_tmp:nn #1 }

```

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

```

383 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
384 {
385   \keys_define:nn { enumext / #1 }
386   {
387     align .choice:,
388     align / left .code:n = \str_set:cn { l__enumext_align_label_#2_str } { l },
389     align / right .code:n = \str_set:cn { l__enumext_align_label_#2_str } { r },

```

```

390     align / center .code:n = \str_set:cn { \__enumext_align_label_#2_str } { c },
391     align .initial:n = left,
392     align .value_required:n = true,
393   }
394 }
395 \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.

```

396 \cs_new_protected:Nn \__enumext_regex_label_ref_key:
397 {
398   \tl_map_inline:Nn \c__enumext_counter_style_tl
399   {
400     \regex_replace_once:nnN { \c{##1}\* }
401     { \c{##1}\cB{\u{\__enumext_ref_aux_tl}\cE} } \__enumext_ref_key_arg_tl
402   }
403 }

```

(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`.

```

404 \cs_new_protected:Npn \__enumext_set_label_ref:n #1
405 {
406   \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
407   \tl_set_eq:Nc \l__enumext_ref_aux_tl { \__enumext_counter_ \__enumext_level: _tl }
408   \__enumext_regex_label_ref_key:
409   \tl_set_eq:Nc \l__enumext_ref_aux_tl { \__enumext_the_counter_ \__enumext_level: _tl }
410   \tl_put_right:ce { \__enumext_counter_style_for_ref_ \__enumext_level: _tl }
411   {
412     \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_aux_tl }
413     { \exp_not:V \l__enumext_ref_key_arg_tl }
414   }
415 }

```

(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`.

```

416 \cs_new_protected:Nn \__enumext_use_key_ref:
417 {
418   \tl_if_empty:cF { \__enumext_counter_style_for_ref_ \__enumext_level: _tl }
419   {
420     \tl_use:c { \__enumext_counter_style_for_ref_ \__enumext_level: _tl }
421   }
422 }

```

(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`.

```

423 \cs_new_protected:Npn \__enumext_set_label_ref_h:n #1
424 {

```

```

425 \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
426 \int_compare:nNnTF { \l__enumext_level_h_int } = { 1 }
427 {
428   \tl_set_eq:NN \l__enumext_ref_aux_tl \l__enumext_counter_vii_tl
429   \__enumext_regex_label_ref_key:
430   \tl_set_eq:NN \l__enumext_ref_aux_tl \l__enumext_the_counter_vii_tl
431   \tl_put_right:Ne \l__enumext_counter_style_for_ref_vii_tl
432   {
433     \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_aux_tl }
434     { \exp_not:V \l__enumext_ref_key_arg_tl }
435   }
436 }
437 {
438   \tl_set_eq:NN \l__enumext_ref_aux_tl \l__enumext_counter_viii_tl
439   \__enumext_regex_label_ref_key:
440   \tl_set_eq:NN \l__enumext_ref_aux_tl \l__enumext_the_counter_viii_tl
441   \tl_put_right:Ne \l__enumext_counter_style_for_ref_viii_tl
442   {
443     \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_aux_tl }
444     { \exp_not:V \l__enumext_ref_key_arg_tl }
445   }
446 }
447 }

```

(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*`.

```

448 \cs_new_protected:Nn \__enumext_use_key_ref_h:
449 {
450   \int_compare:nNnTF { \l__enumext_level_h_int } = { 1 }
451   {
452     \tl_if_empty:NF \l__enumext_counter_style_for_ref_vii_tl
453     {
454       \tl_use:N \l__enumext_counter_style_for_ref_vii_tl
455     }
456   }
457   {
458     \tl_if_empty:NF \l__enumext_counter_style_for_ref_viii_tl
459     {
460       \tl_use:N \l__enumext_counter_style_for_ref_viii_tl
461     }
462   }
463 }

```

(End of definition for `__enumext_use_key_ref_h:`.)

10.11.1 Define and set label key for enumext environment

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

```

\l__enumext_label_i_tl 464 \cs_set_protected:Npn \__enumext_tmp:nnn #1 #2 #3
\l__enumext_label_ii_tl 465 {
\l__enumext_label_iii_tl 466   \keys_define:nn { enumext / #1 }
\l__enumext_label_iv_tl 467   {
468     label .code:n = {
469       \__enumext_label_style:cvn { l__enumext_label_#2_tl }
470       { l__enumext_counter_#2_tl } {##1}
471       \dim_set_eq:cN { l__enumext_labelwidth_#2_dim }
472       \l__enumext_current_widest_dim
473     },
474     label .initial:n = #3,
475     label .value_required:n = true,
476     ref .code:n = \__enumext_set_label_ref:n {##1},
477     ref .value_required:n = true,
478   }
479 }
480 \__enumext_tmp:nnn { level-1 } { i } { \arabic*. }
481 \__enumext_tmp:nnn { level-2 } { ii } { (\alph*) }
482 \__enumext_tmp:nnn { level-3 } { iii } { \roman*. }
483 \__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 $\langle label \rangle$ for `enumext*` and `keyans*` environments, along with the default value for `labelwidth` key.

```

label
ref
\l__enumext_label_vii_tl
\l__enumext_label_viii_tl
484 \cs_set_protected:Npn \__enumext_tmp:nnn #1 #2 #3
485 {
486   \keys_define:nn { enumext / #1 }
487   {
488     label .code:n = {
489       \__enumext_label_style:cvn { \l__enumext_label_#2_tl }
490       { \l__enumext_counter_#2_tl } {##1}
491       \dim_set_eq:cN { \l__enumext_labelwidth_#2_dim }
492       \l__enumext_current_widest_dim
493     },
494     label .initial:n = #3,
495     label .value_required:n = true,
496     ref .code:n = \__enumext_set_label_ref_h:n {##1},
497     ref .value_required:n = true,
498   }
499 }
500 \__enumext_tmp:nnn { enumext* } { vii } { \arabic*.}
501 \__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 $\langle label \rangle$ for `keyans` and `keyanspic` environment, along with the default value for `labelwidth`. The `keyanspic` environment use the same $\langle label \rangle$ as the `keyans` environment.

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

```

502 \keys_define:nn { enumext / keyans }
503 {
504   label .code:n = {
505     \__enumext_label_style:cvn { \l__enumext_label_v_tl }
506     { \l__enumext_counter_v_tl } {##1}
507     \dim_set_eq:cN { \l__enumext_labelwidth_v_dim }
508     \l__enumext_current_widest_dim
509     \__enumext_label_style:cvn { \l__enumext_label_vi_tl }
510     { \l__enumext_counter_vi_tl } {##1}
511     \dim_set_eq:cN { \l__enumext_labelwidth_v_dim }
512     \l__enumext_current_widest_dim
513   },
514   label .initial:n = (\Alph*),
515   label .value_required:n = true,
516 }

```

(End of definition for `label`, `\l__enumext_label_v_tl`, and `\l__enumext_label_vi_tl`.)

10.12 Setting start and widest keys

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

```

\__enumext_start_from:NNn
\__enumext_start_from:ccn
#1: \l__enumext_label_X_tl
#2: \l__enumext_start_X_int
#3:  $\langle integer \text{ or } string \rangle$ 

```

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 $\langle integer \rangle$ or $\langle string \rangle$ of the form `\Alph`, `\alph`, `\Roman` or `\roman`. This effectively allows `start=A` or `start=1` to be used.

```

517 \cs_new_protected:Npn \__enumext_start_from:NNn #1 #2 #3
518 {
519   \__enumext_if_is_int:nTF { #3 }
520   {
521     \int_set:Nn #2 {##3}
522   }
523   {
524     \regex_match:nVT { \c{Alph} | \c{alph} } {##1}
525     { \int_set:Nn #2 { \int_from_alph:n {##3} } }
526     \regex_match:nVT { \c{Roman} | \c{roman} } {##1}
527     { \int_set:Nn #2 { \int_from_roman:n {##3} } }
528   }
529 }
530 \cs_generate_variant:Nn \__enumext_start_from:NNn { ccn }

```

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

`__enumext_widest_from:nNNn`
`__enumext_widest_from:nccn`

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

- #1: The counter associated with the environment level
- #2: `\l__enumext_label_X_tl`
- #3: `\l__enumext_labelwidth_X_dim`
- #4: $\langle integer \text{ or } string \rangle$

The second and third arguments of this function are the values set by `label` and `labelwidth` keys, the four argument can be an $\langle integer \rangle$ or $\langle string \rangle$ 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.

```

531 \cs_new_protected:Npn \__enumext_widest_from:nNNn #1 #2 #3 #4
532 {
533   \__enumext_if_is_int:nTF {#4}
534   {
535     \setcounter{enumX#1} { #4 }
536   }
537   {
538     \regex_match:nVT { \c{Alph} | \c{alph} } {#2}
539     { \setcounter{enumX#1} { \int_from_alph:n {#4} } }
540     \regex_match:nVT { \c{Roman} | \c{roman} } {#2}
541     { \setcounter{enumX#1} { \int_from_roman:n {#4} } }
542   }
543   \__enumext_label_width_by_box:cv
544   { \l__enumext_labelwidth_#1_dim } { \l__enumext_label_#1_tl }
545 }
546 \cs_generate_variant:Nn \__enumext_widest_from:nNNn { nccn }

```

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

`start`
`widest`

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

`\l__enumext_start_X_int`

```

547 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
548 {
549   \keys_define:nn { enumext / #1 }
550   {
551     start .code:n = {
552       \__enumext_start_from:ccn
553       { \l__enumext_label_#2_tl }
554       { \l__enumext_start_#2_int } {##1}
555     },
556     start .initial:n = 1,
557     widest .code:n = {
558       \__enumext_widest_from:nccn {#2}
559       { \l__enumext_label_#2_tl }
560       { \l__enumext_labelwidth_#2_dim } {##1}
561     },
562     widest .value_required:n = true,
563     start .value_required:n = true,
564   }
565 }
566 \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

`topsep`
`partopsep`
`parsep`
`noitemsep`
`nosep`

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

```

567 \cs_set_protected:Npn \__enumext_tmp:nnnnn #1 #2 #3 #4 #5 #6
568 {
569   \keys_define:nn { enumext / #1 }
570   {
571     topsep .skip_set:c = { \l__enumext_topsep_#2_skip },
572     topsep .initial:n = {#3},
573     topsep .value_required:n = true,
574     partopsep .skip_set:c = { \l__enumext_partopsep_#2_skip },
575     partopsep .initial:n = {#4},
576     partopsep .value_required:n = true,
577     parsep .skip_set:c = { \l__enumext_parsep_#2_skip },
578     parsep .initial:n = {#5},

```

```

579     parsep      .value_required:n = true,
580     itemsep     .skip_set:c = { l__enumext_itemsep_#2_skip },
581     itemsep     .initial:n = {#6},
582     itemsep     .value_required:n = true,
583     noitemsep   .meta:n = { itemsep = 0pt, parsep = 0pt },
584     noitemsep   .value_forbidden:n = true,
585     nosepe      .meta:n = {
586                         itemsep = 0pt, parsep= 0pt,
587                         topsep = 0pt, partopsep = 0pt,
588                     },
589     nosepe      .value_forbidden:n = true,
590 }
591 }

```

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

```

592 \__enumext_tmp:nnnnnn { level-1 } { i } { 8.0pt plus 2.0pt minus 4.0pt }
593 { 2.0pt plus 1.0pt minus 1.0pt } { 4.0pt plus 2.0pt minus 1.0pt }
594 { 4.0pt plus 2.0pt minus 1.0pt }
595 \__enumext_tmp:nnnnnn { level-2 } { ii } { 4.0pt plus 2.0pt minus 1.0pt }
596 { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
597 { 2.0pt plus 1.0pt minus 1.0pt }
598 \__enumext_tmp:nnnnnn { level-3 } { iii } { 2.0pt plus 1.0pt minus 1.0pt }
599 { 1.0pt minus 1.0pt } { 0pt } { 2.0pt plus 1.0pt minus 1.0pt }
600 \__enumext_tmp:nnnnnn { level-4 } { iv } { 2.0pt plus 1.0pt minus 1.0pt }
601 { 1.0pt minus 1.0pt } { 0pt } { 2.0pt plus 1.0pt minus 1.0pt }
602 \__enumext_tmp:nnnnnn { keyans } { v } { 4.0pt plus 2.0pt minus 1.0pt }
603 { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
604 { 2.0pt plus 1.0pt minus 1.0pt }
605 \__enumext_tmp:nnnnnn { enumext* } { vii } { 8.0pt plus 2.0pt minus 4.0pt }
606 { 2.0pt plus 1.0pt minus 1.0pt } { 4.0pt plus 2.0pt minus 1.0pt }
607 { 4.0pt plus 2.0pt minus 1.0pt }
608 \__enumext_tmp:nnnnnn { keyans* } { viii } { 4.0pt plus 2.0pt minus 1.0pt }
609 { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
610 { 2.0pt plus 1.0pt minus 1.0pt }

```

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

10.14 Setting keys for horizontal spaces

Define and set `itemindent`, `rightmargin`, `listparindent`, `list-offset` and `list-indent` keys for `enumext` and `keyans` environments.

```

611 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
612 {
613     \keys_define:nn { enumext / #1 }
614     {
615         itemindent      .dim_set:c = { l__enumext_fake_item_indent_#2_dim },
616         itemindent      .value_required:n = true,
617         rightmargin     .dim_set:c = { l__enumext_rightmargin_#2_dim },
618         rightmargin     .value_required:n = true,
619         listparindent   .dim_set:c = { l__enumext_listparindent_#2_dim },
620         listparindent   .value_required:n = true,
621         list-offset     .dim_set:c = { l__enumext_listoffset_#2_dim },
622         list-offset     .value_required:n = true,
623         list-indent     .code:n =
624                         \bool_set_true:c { l__enumext_leftmargin_tmp_#2_bool }
625                         \dim_set:cn { l__enumext_leftmargin_tmp_#2_dim } {##1},
626         list-indent     .value_required:n = true,
627     }
628 }
629 \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.

```

630 \cs_set_protected:Npn \__enumext_tmp:n #1
631 {
632     \keys_define:nn { enumext / #1 } { list-indent .initial:n = 0pt, }
633 }
634 \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 `\opt`. 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.

```

635 \cs_set_protected:Nn \__enumext_fake_item:
636 {
637   \dim_compare:nNnT
638     { \dim_use:c { l__enumext_fake_item_indent_ \__enumext_level: _dim } }
639     >
640     { \c_zero_dim }
641   {
642     \tl_set:ce { l__enumext_fake_item_indent_ \__enumext_level: _tl }
643     {
644       \exp_not:N \mode_leave_vertical:
645       \exp_not:n { \skip_horizontal:n }
646       { \dim_use:c { l__enumext_fake_item_indent_ \__enumext_level: _dim } }
647       \ignorespaces
648     }
649   }
650 }
651 \cs_set_protected:Nn \__enumext_keyans_fake_item:
652 {
653   \dim_compare:nNnT
654     { \l__enumext_fake_item_indent_v_dim } > { \c_zero_dim }
655   {
656     \tl_set:Ne \l__enumext_fake_item_indent_v_tl
657     {
658       \exp_not:N \mode_leave_vertical:
659       \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_v_dim
660     }
661   }
662 }
663 \cs_set_protected:Nn \__enumext_fake_item_vii:
664 {
665   \dim_compare:nNnT
666     { \l__enumext_fake_item_indent_vii_dim } > { \c_zero_dim }
667   {
668     \tl_set:Ne \l__enumext_fake_item_indent_vii_tl
669     {
670       \exp_not:N \mode_leave_vertical:
671       \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_vii_dim
672     }
673   }
674 }
675 \cs_set_protected:Nn \__enumext_fake_item_viii:
676 {
677   \dim_compare:nNnT
678     { \l__enumext_fake_item_indent_viii_dim } > { \c_zero_dim }
679   {
680     \tl_set:Ne \l__enumext_fake_item_indent_viii_tl
681     {
682       \exp_not:N \mode_leave_vertical:
683       \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_viii_dim
684     }
685   }
686 }

```

(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.

```

687 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
688 {
689   \keys_define:nn { enumext / #1 }
690   {
691     show-length .bool_set:c = { l__enumext_show_length_#2_bool },

```

```

692         show-length .initial:n = false,
693     }
694 }
695 \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

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

```

before* 696 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
after    697 {
first    698     \keys_define:nn { enumext / #1 }
        699     {
        700         before .tl_set:c = { l__enumext_before_no_starred_key_#2_tl },
        701         before .value_required:n = true,
        702         before* .tl_set:c = { l__enumext_before_starred_key_#2_tl },
        703         before* .value_required:n = true,
        704         after .tl_set:c = { l__enumext_after_stop_list_#2_tl },
        705         after .value_required:n = true,
        706         first .tl_set:c = { l__enumext_after_list_args_#2_tl },
        707         first .value_required:n = true,
        708     }
        709 }
        710 \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 $\{\langle code \rangle\}$ set by the `before*` key “before” the `enumext` environment is started. The $\{\langle code \rangle\}$ is executed “without” knowing any definition of the *second argument* of the list.

```

__enumext_before_args_exec:
__enumext_before_keys_exec:
__enumext_after_stop_list:
__enumext_after_args_exec:
711 \cs_new_protected:Nn \__enumext_before_args_exec:
712 {
713     \tl_use:c { l__enumext_before_starred_key_ \__enumext_level: _tl }
714 }

```

The function `__enumext_before_keys_exec:` executes the $\{\langle code \rangle\}$ set by the `before` key “before” the `enumext` environment is started in *second argument* of the list. The $\{\langle code \rangle\}$ is executed “knowing” all definition and values provides by $\langle keys \rangle$.

```

715 \cs_new_protected:Nn \__enumext_before_keys_exec:
716 {
717     \tl_use:c { l__enumext_before_no_starred_key_ \__enumext_level: _tl }
718 }

```

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

```

719 \cs_new_protected:Nn \__enumext_after_stop_list:
720 {
721     \tl_use:c { l__enumext_after_stop_list_ \__enumext_level: _tl }
722 }

```

The function `__enumext_after_args_exec:` executes the $\{\langle code \rangle\}$ 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 $\backslash item$.

```

723 \cs_new_protected:Nn \__enumext_after_args_exec:
724 {
725     \tl_use:c { l__enumext_after_list_args_ \__enumext_level: _tl }
726 }

```

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

10.16.2 Functions for before, after and first keys in keyans

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

```

__enumext_before_args_exec_v:
__enumext_before_keys_exec_v:
__enumext_after_stop_list_v:
__enumext_after_args_exec_v:
727 \cs_new_protected:Nn \__enumext_before_args_exec_v:
728 {
729     \tl_use:N \l__enumext_before_starred_key_v_tl
730 }

```

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⟩`.

```
731 \cs_new_protected:Nn \__enumext_before_keys_exec_v:
732 {
733   \tl_use:N \l__enumext_before_no_starred_key_v_tl
734 }
```

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

```
735 \cs_new_protected:Nn \__enumext_after_stop_list_v:
736 {
737   \tl_use:N \l__enumext_after_stop_list_v_tl
738 }
```

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`.

```
739 \cs_new_protected:Nn \__enumext_after_args_exec_v:
740 {
741   \tl_use:N \l__enumext_after_list_args_v_tl
742 }
```

(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.

```
743 \cs_new_protected:Nn \__enumext_before_args_exec_vii:
744 {
745   \tl_use:N \l__enumext_before_starred_key_vii_tl
746 }
747 \cs_new_protected:Nn \__enumext_before_args_exec_viii:
748 {
749   \tl_use:N \l__enumext_before_starred_key_viii_tl
750 }
```

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⟩`.

```
751 \cs_new_protected:Nn \__enumext_before_keys_exec_vii:
752 {
753   \tl_use:N \l__enumext_before_no_starred_key_vii_tl
754 }
755 \cs_new_protected:Nn \__enumext_before_keys_exec_viii:
756 {
757   \tl_use:N \l__enumext_before_no_starred_key_viii_tl
758 }
```

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

```
759 \cs_new_protected:Nn \__enumext_after_stop_list_vii:
760 {
761   \tl_use:N \l__enumext_after_stop_list_vii_tl
762 }
763 \cs_new_protected:Nn \__enumext_after_stop_list_viii:
764 {
765   \tl_use:N \l__enumext_after_stop_list_viii_tl
766 }
```

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`.

```
767 \cs_new_protected:Nn \__enumext_after_args_exec_vii:
768 {
769   \tl_use:N \l__enumext_after_list_args_vii_tl
770 }
771 \cs_new_protected:Nn \__enumext_after_args_exec_viii:
772 {
773   \tl_use:N \l__enumext_after_list_args_viii_tl
774 }
```

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

10.17 Setting keys for multicols 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

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

columns

```

775 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
776 {
777   \keys_define:nn { enumext / #1 }
778   {
779     mini-env .dim_set:c = { \__enumext_minipage_right_#2_dim },
780     mini-env .value_required:n = true,
781     mini-sep .dim_set:c = { \__enumext_minipage_hsep_#2_dim },
782     mini-sep .initial:n = 0.3333em,
783     mini-sep .value_required:n = true,
784     columns-sep .dim_set:c = { \__enumext_columns_sep_#2_dim },
785     columns-sep .value_required:n = true,
786     columns .int_set:c = { \__enumext_columns_#2_int },
787     columns .initial:n = 1,
788     columns .value_required:n = true,
789   }
790 }
791 \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`.

```

792 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
793 {
794   \keys_define:nn { enumext / #1 }
795   {
796     columns .initial:n = 2,
797     miniright .tl_gset:c = { g__enumext_miniright_code_#2_tl },
798     miniright .value_required:n = true,
799     miniright* .code:n = {
800       \bool_gset_true:c { g__enumext_minipage_center_#2_bool }
801       \keys_set:nn { enumext / #1 } { miniright = {##1} }
802     },
803     miniright* .value_required:n = true,
804   }
805 }
806 \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 multicols

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

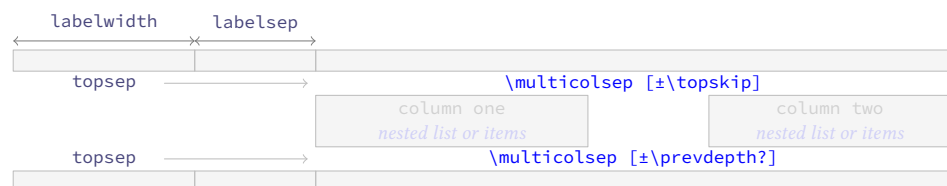


Figure 7: Representation of the vertical space in `multicols` for a nested level.

To keep the desired spaces *above* and *below* in the “*list environment*” (`\topsep` + `[\partopsep]`) it is necessary to “*adjust*” the spaces added by the `multicols` environment. The most appropriate option in this case is to use a “*context sensitive*” vertical space with `\addvspace`.

- I should make it clear that the implementation here is a “*bit questionable*”. At first glance doing `\multicolsep=\topsep` seemed right, but the results were not always as expected. An almost *imperceptible* detail is that in some cases the `\itemsep` values of are “*stretched*”, possibly due to the use of `\raggedcolumns` and this affects the lower space when closing the environment, which is “*smaller*” than expected. My attempts to find the correct values using `\showoutput` and `\showboxdepth` absolutely failed.

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 $\langle\textit{horizontal mode}\rangle$, then we will make the settings for the $\langle\textit{vertical mode}\rangle$ in which `\partopsep` comes into play.

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

```

807 \cs_new_protected:Nn \__enumext_multi_set_vskip:
808 {
809   \skip_set:cn { \l__enumext_multicols_above_ \__enumext_level: _skip }
810   {
811     \skip_use:c { \l__enumext_topsep_ \__enumext_level: _skip }
812   }
813   \skip_set:cn { \l__enumext_multicols_below_ \__enumext_level: _skip }
814   {
815     \skip_use:c { \l__enumext_topsep_ \__enumext_level: _skip }
816   }
817   \__enumext_add_pre_parsep:
818 }

```

(End of definition for `__enumext_multi_set_vskip:`.)

`__enumext_add_pre_parsep:` The function `__enumext_add_pre_parsep:` “*adjusted*” the value of `\l__enumext_multicols_above_X_skip` detecting the value of `\parsep` from the previous level. This is necessary since `\parsep` from the previous level affects the *vertical spaces*.

```

819 \cs_new_protected:Nn \__enumext_add_pre_parsep:
820 {
821   \int_case:nn { \l__enumext_level_int }
822   {
823     { 2 }{
824       \skip_if_eq:nnF { \l__enumext_parsep_i_skip } { \c_zero_skip }
825       {
826         \skip_add:Nn \l__enumext_multicols_above_ii_skip { \l__enumext_parsep_i_skip }
827       }
828     }
829     { 3 }{
830       \skip_if_eq:nnF { \l__enumext_parsep_ii_skip } { \c_zero_skip }
831       {
832         \skip_add:Nn \l__enumext_multicols_above_iii_skip { \l__enumext_parsep_ii_skip }
833       }
834     }
835     { 4 }{
836       \skip_if_eq:nnF { \l__enumext_parsep_iii_skip } { \c_zero_skip }
837       {
838         \skip_add:Nn \l__enumext_multicols_above_iv_skip { \l__enumext_parsep_iii_skip }
839       }
840     }
841   }
842 }

```

(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 `multicol` environment in `enumext`, taking into account whether \TeX is in $\langle\textit{horizontal mode}\rangle$ or $\langle\textit{vertical mode}\rangle$.

```

843 \cs_new_protected:Nn \__enumext_multi_addvspace:
844 {
845   \__enumext_multi_set_vskip:
846   \mode_if_vertical:T
847   {
848     \skip_add:cn { \l__enumext_multicols_above_ \__enumext_level: _skip }
849     {
850       \skip_use:c { \l__enumext_partopsep_ \__enumext_level: _skip }
851     }
852     \skip_add:cn { \l__enumext_multicols_below_ \__enumext_level: _skip }
853     {
854       \skip_use:c { \l__enumext_partopsep_ \__enumext_level: _skip }
855     }
856   }

```

```

857 \par\nopagebreak
858 \addvspace{ \skip_use:c { \l__enumext_multicols_above_ \l__enumext_level: \skip } }
859 }

```

(End of definition for `\l__enumext_multi_addvspace:`.)

10.18.2 Adjustment of vertical spaces for multicols in keyans

`\l__enumext_keyans_multi_set_vskip:` The function `\l__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`.

```

860 \cs_new_protected:Nn \l__enumext_keyans_multi_set_vskip:
861 {
862   \skip_set:Nn \l__enumext_multicols_above_v_skip
863   {
864     \l__enumext_topsep_v_skip
865   }
866   \skip_set:Nn \l__enumext_multicols_below_v_skip
867   {
868     \l__enumext_topsep_v_skip
869   }
870 }
871 \cs_new_protected:Nn \l__enumext_keyans_multi_addvspace:
872 {
873   \l__enumext_keyans_multi_set_vskip:
874   \mode_if_vertical:T
875   {
876     \skip_add:Nn \l__enumext_multicols_above_v_skip
877     {
878       \skip_use:N \l__enumext_partopsep_v_skip
879     }
880     \skip_add:Nn \l__enumext_multicols_below_v_skip
881     {
882       \skip_use:N \l__enumext_partopsep_v_skip
883     }
884   }
885   \par\nopagebreak
886   \addvspace{ \l__enumext_multicols_above_v_skip }
887 }

```

(End of definition for `\l__enumext_keyans_multi_set_vskip:` and `\l__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.

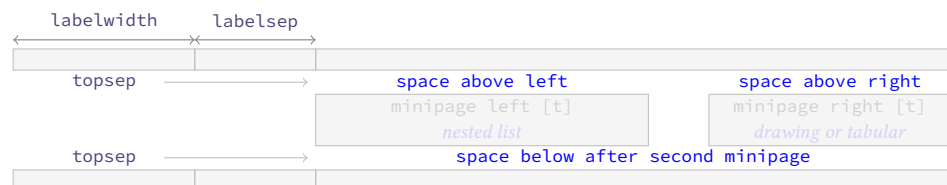


Figure 8: Representation of the `minipage` spacing adjustment for a nested level.

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.

`\l__enumext_mini_env*` Creates a `\l__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.

```

888 \DeclareDocumentEnvironment{__enumext_mini_env*}{ m }
889 {
890     \__enumext_minipage:w [ t ] { #1 }
891     \legacy_if_gset_false:n { @minipage }
892     \vspace { 0pt }
893 }
894 { \__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 T_EX 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`.

```

895 \cs_new_protected:Nn \__enumext_mini_set_vskip:
896 {
897     \int_compare:nNnTF
898     { \int_use:c { \l__enumext_columns_ \__enumext_level: _int } } > { 1 }
899     {

```

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.

```

900     \skip_if_eq:nnTF
901     { \skip_use:c { \l__enumext_topsep_ \__enumext_level: _skip } } { \c_zero_skip }
902     {
903         \skip_set:Nn \l__enumext_minipage_left_skip
904         {
905             -0.150\box_dp:N \strutbox
906         }
907         \skip_set:Nn \l__enumext_minipage_right_skip
908         {
909             0.695\box_dp:N \strutbox
910         }
911         \skip_set:Nn \l__enumext_minipage_after_skip
912         {
913             \box_dp:N \strutbox
914         }
915         \__enumext_zero_parsep:
916     }
917     {
918         \skip_set:Nn \l__enumext_minipage_left_skip
919         {
920             \skip_use:c { \l__enumext_topsep_ \__enumext_level: _skip }
921         }
922         \skip_set:Nn \l__enumext_minipage_right_skip
923         {
924             0.695\box_dp:N \strutbox
925         }
926         \skip_set:Nn \l__enumext_minipage_after_skip
927         {
928             1.85\box_dp:N \strutbox
929             + \skip_use:c { \l__enumext_topsep_ \__enumext_level: _skip }
930         }
931     }
932 }
933 {

```

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.

```

934     \skip_if_eq:nnTF
935     { \skip_use:c { \l__enumext_topsep_ \__enumext_level: _skip } } { \c_zero_skip }
936     {
937         \skip_set:Nn \l__enumext_minipage_left_skip

```

```

938         {
939             0.5\box_dp:N \strutbox
940             - \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
941         }
942     \skip_set:Nn \l__enumext_minipage_right_skip
943     {
944         \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
945     }
946     \skip_set:Nn \l__enumext_minipage_after_skip
947     {
948         1.6\box_dp:N \strutbox
949     }
950 }
951 {
952     \skip_set:Nn \l__enumext_minipage_left_skip
953     {
954         0.5875\box_dp:N \strutbox
955         - \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
956     }
957     \skip_set:Nn \l__enumext_minipage_right_skip
958     {
959         + \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
960         + \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
961     }
962     \skip_set:Nn \l__enumext_minipage_after_skip
963     {
964         0.325\box_dp:N \strutbox
965         + \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
966     }
967 }
968 }
969 }

```

(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.

```

970 \cs_new_protected:Nn \__enumext_zero_parsep:
971 {
972     \int_case:nn { \l__enumext_level_int }
973     {
974         { 2 }{
975             \skip_if_eq:nnF { \l__enumext_parsep_i_skip } { \c_zero_skip }
976             {
977                 \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
978             }
979         }
980         { 3 }{
981             \skip_if_eq:nnF { \l__enumext_parsep_ii_skip } { \c_zero_skip }
982             {
983                 \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
984             }
985         }
986         { 4 }{
987             \skip_if_eq:nnF { \l__enumext_parsep_iii_skip } { \c_zero_skip }
988             {
989                 \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
990             }
991         }
992     }
993 }

```

(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 T_EX 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*.

```

994 \cs_new_protected:Nn \__enumext_mini_addvspace:
995 {

```

```

996 \__enumext_mini_set_vskip:
997 \mode_if_vertical:T
998 {
999   \skip_add:Nn \l__enumext_minipage_left_skip
1000   {
1001     \skip_use:c { \l__enumext_partopsep_ \__enumext_level: _skip }
1002   }
1003   \skip_add:Nn \l__enumext_minipage_after_skip
1004   {
1005     \skip_use:c { \l__enumext_partopsep_ \__enumext_level: _skip }
1006   }
1007 }
1008 \par\nopagebreak
1009 \addvspace { \l__enumext_minipage_left_skip }
1010 }

```

(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`.

```

1011 \cs_new_protected:Nn \__enumext_keyans_mini_set_vskip:
1012 {
1013   \skip_zero_new:N \l__enumext_minipage_after_skip
1014   \skip_zero_new:N \l__enumext_minipage_left_skip
1015   \skip_zero_new:N \l__enumext_minipage_right_skip
1016   \int_compare:nNnTF { \l__enumext_columns_v_int } > { 1 }
1017   {
1018     \skip_if_eq:nnTF { \l__enumext_topsep_v_skip } { \c_zero_skip }
1019     {
1020       \skip_set:Nn \l__enumext_minipage_left_skip { -0.25\box_dp:N \strutbox }
1021       \skip_set:Nn \l__enumext_minipage_right_skip { 0.705\box_dp:N \strutbox }
1022       \skip_set:Nn \l__enumext_minipage_after_skip { \box_dp:N \strutbox }
1023       \skip_if_eq:nnF { \l__enumext_parsep_i_skip } { \c_zero_skip }
1024       {
1025         \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
1026       }
1027     }
1028     {
1029       \skip_set:Nn \l__enumext_minipage_left_skip
1030       {
1031         \skip_use:N \l__enumext_topsep_v_skip
1032       }
1033       \skip_set:Nn \l__enumext_minipage_right_skip
1034       {
1035         0.705\box_dp:N \strutbox
1036       }
1037       \skip_set:Nn \l__enumext_minipage_after_skip
1038       {
1039         1.85\box_dp:N \strutbox + \l__enumext_topsep_v_skip
1040       }
1041     }
1042   }
1043   {
1044     \skip_if_eq:nnTF { \l__enumext_topsep_v_skip } { \c_zero_skip }
1045     {
1046       \skip_set:Nn \l__enumext_minipage_left_skip
1047       {
1048         0.5\box_dp:N \strutbox
1049         + \l__enumext_partopsep_v_skip
1050       }
1051       \skip_set:Nn \l__enumext_minipage_right_skip
1052       {
1053         \l__enumext_partopsep_v_skip
1054       }
1055       \skip_set:Nn \l__enumext_minipage_after_skip { 1.6\box_dp:N \strutbox }
1056     }
1057     {
1058       \skip_set:Nn \l__enumext_minipage_left_skip
1059       {

```

```

1060         0.5875\box_dp:N \strutbox - \l__enumext_partopsep_v_skip
1061     }
1062     \skip_set:Nn \l__enumext_minipage_right_skip
1063     {
1064         \l__enumext_topsep_v_skip + \l__enumext_partopsep_v_skip
1065     }
1066     \skip_set:Nn \l__enumext_minipage_after_skip
1067     {
1068         0.325\box_dp:N \strutbox + \l__enumext_topsep_v_skip
1069     }
1070 }
1071 }
1072 }

```

(End of definition for `__enumext_keyans_mini_set_vskip:`.)

`__enumext_keyans_mini_addvspace:`

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

```

1073 \cs_new_protected:Nn \__enumext_keyans_mini_addvspace:
1074 {
1075     \__enumext_keyans_mini_set_vskip:
1076     \mode_if_vertical:T
1077     {
1078         \skip_add:Nn \l__enumext_minipage_left_skip
1079         {
1080             \l__enumext_partopsep_v_skip
1081         }
1082         \skip_add:Nn \l__enumext_minipage_after_skip
1083         {
1084             \l__enumext_partopsep_v_skip
1085         }
1086     }
1087     \par\nopagebreak
1088     \addvspace { \l__enumext_minipage_left_skip }
1089 }

```

(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*`.

```

1090 \cs_new_protected:Nn \__enumext_mini_set_vskip_vii:
1091 {
1092     \skip_zero_new:N \l__enumext_minipage_left_skip
1093     \skip_gzero_new:N \g__enumext_minipage_right_skip
1094     \skip_gzero_new:N \g__enumext_minipage_after_skip
1095     \skip_if_eq:nnTF { \l__enumext_topsep_vii_skip } { \c_zero_skip }
1096     {
1097         \skip_set:Nn \l__enumext_minipage_left_skip { 0.5\box_dp:N \strutbox }
1098         \skip_gset:Nn \g__enumext_minipage_right_skip { 0.325\box_dp:N \strutbox }
1099     }
1100     {
1101         \skip_set:Nn \l__enumext_minipage_left_skip { 0.5875\box_dp:N \strutbox }
1102         \skip_gset:Nn \g__enumext_minipage_right_skip
1103         {
1104             \l__enumext_topsep_vii_skip
1105         }
1106         \skip_gset:Nn \g__enumext_minipage_after_skip
1107         {
1108             0.325\box_dp:N \strutbox + \l__enumext_topsep_vii_skip
1109         }
1110     }
1111 }
1112 \cs_new_protected:Nn \__enumext_mini_set_vskip_viii:
1113 {
1114     \skip_zero_new:N \l__enumext_minipage_after_skip

```

```

1115 \skip_zero_new:N \l__enumext_minipage_left_skip
1116 \skip_zero_new:N \l__enumext_minipage_right_skip
1117 \skip_if_eq:nnTF { \l__enumext_topsep_viii_skip } { \c_zero_skip }
1118 {
1119   \skip_set:Nn \l__enumext_minipage_left_skip
1120   {
1121     0.5\box_dp:N \strutbox
1122   }
1123   \skip_set:Nn \l__enumext_minipage_right_skip
1124   {
1125     \l__enumext_partopsep_viii_skip
1126   }
1127   \skip_set:Nn \l__enumext_minipage_after_skip
1128   {
1129     1.6\box_dp:N \strutbox
1130   }
1131 }
1132 {
1133   \skip_set:Nn \l__enumext_minipage_left_skip
1134   {
1135     0.5875\box_dp:N \strutbox
1136   }
1137   \skip_set:Nn \l__enumext_minipage_right_skip
1138   {
1139     \l__enumext_topsep_viii_skip
1140   }
1141   \skip_set:Nn \l__enumext_minipage_after_skip
1142   {
1143     0.325\box_dp:N \strutbox + \l__enumext_topsep_viii_skip
1144   }
1145 }
1146 }

```

(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 \TeX is in *horizontal mode* or *vertical mode*, since `\partopsep` is equal to `0pt` in both environments.

```

1147 \cs_new_protected:Nn \__enumext_mini_addvspace_vii:
1148 {
1149   \__enumext_mini_set_vskip_vii:
1150   \par\nopagebreak
1151   \addvspace { \l__enumext_minipage_left_skip }
1152 }
1153 \cs_new_protected:Nn \__enumext_mini_addvspace_viii:
1154 {
1155   \__enumext_mini_set_vskip_viii:
1156   \par\nopagebreak
1157   \addvspace { \l__enumext_minipage_left_skip }
1158 }

```

(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 \TeX 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.

```

1159 \NewDocumentCommand \miniright { s }
1160 {
1161   \int_compare:nNt { \l__enumext_keyans_pic_level_int } = { 1 }
1162   {
1163     \msg_error:nnn { enumext } { wrong-miniright-place }

```



```

1164     }
1165     \int_compare:nNt { \__enumext_level_int } = { 0 }
1166     {
1167         \msg_error:nnn { enumext } { wrong-miniright-place }
1168     }
1169     \int_compare:nNtF { \__enumext_keyans_level_int } = { 1 }
1170     {
1171         \__enumext_keyans_mini_right_cmd:n {#1}
1172     }
1173     { \__enumext_mini_right_cmd:n {#1} }
1174 }

```

(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.

```

1175 \cs_new_protected:Npn \__enumext_mini_right_cmd:n #1
1176 {
1177     \dim_compare:nNtF
1178     { \dim_use:c { \__enumext_minipage_right_ \__enumext_level: _dim } } > { \c_zero_dim }
1179     {
1180         \__enumext_multicols_stop:
1181         \end{__enumext_mini_env*}
1182         \hfill
1183         \begin{__enumext_mini_env*}
1184             { \dim_use:c { \__enumext_minipage_right_ \__enumext_level: _dim } }
1185             \par\addvspace { \__enumext_minipage_right_skip }
1186             \bool_if:nF {#1}
1187             {
1188                 \centering
1189             }
1190             \int_gzero:N \g__enumext_minipage_stat_int
1191         }
1192         { \msg_error:nnn { enumext } { wrong-miniright-use } }
1193     }

```

(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 `\miniright` command in the `keyans` environment. The implementation of this function is the same as that of the `__enumext_mini_right_cmd:n` function of the `enumext` environment.

```

1194 \cs_new_protected:Npn \__enumext_keyans_mini_right_cmd:n #1
1195 {
1196     \dim_compare:nNtF { \__enumext_minipage_right_v_dim } > { \c_zero_dim }
1197     {
1198         \__enumext_keyans_multicols_stop:
1199         \end{__enumext_mini_env*}
1200         \hfill
1201         \begin{__enumext_mini_env*}{ \__enumext_minipage_right_v_dim }
1202             \par\addvspace { \__enumext_minipage_right_skip }
1203             \bool_if:nF {#1}
1204             {
1205                 \centering
1206             }
1207             \int_gzero:N \g__enumext_minipage_stat_int
1208         }
1209         { \msg_error:nnn { enumext } { wrong-miniright-use } }
1210     }

```

(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.

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

```

above* 1211 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
below 1212 {
below* 1213 \keys_define:nn { enumext / #1 }
1214 {
1215     above .skip_set:c = { l__enumext_vspace_above_#2_skip },
1216     above .value_required:n = true,
1217     above* .code:n = \bool_set_true:c { l__enumext_vspace_a_star_#2_bool }
1218             \keys_set:nn { enumext / #1 } { above = {##1} },
1219     above* .value_required:n = true,
1220     below .skip_set:c = { l__enumext_vspace_below_#2_skip },
1221     below .value_required:n = true,
1222     below* .code:n = \bool_set_true:c { l__enumext_vspace_b_star_#2_bool }
1223             \keys_set:nn { enumext / #1 } { below = {##1} },
1224     below* .value_required:n = true,
1225 }
1226 }
1227 \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.

```

1228 \cs_new_protected:Nn \__enumext_vspace_above:
1229 {
1230     \skip_if_eq:nnF
1231     { \skip_use:c { l__enumext_vspace_above_ \__enumext_level: _skip } } { \c_zero_skip }
1232     {
1233         \bool_if:cTF { l__enumext_vspace_a_star_ \__enumext_level: _bool }
1234         {
1235             \vspace*{ \skip_use:c { l__enumext_vspace_above_ \__enumext_level: _skip } }
1236         }
1237         {
1238             \vspace { \skip_use:c { l__enumext_vspace_above_ \__enumext_level: _skip } }
1239         }
1240     }
1241 }

```

(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.

```

1242 \cs_new_protected:Nn \__enumext_vspace_below:
1243 {
1244     \skip_if_eq:nnF
1245     { \skip_use:c { l__enumext_vspace_below_ \__enumext_level: _skip } } { \c_zero_skip }
1246     {
1247         \bool_if:cTF { l__enumext_vspace_b_star_ \__enumext_level: _bool }
1248         {
1249             \vspace*{ \skip_use:c { l__enumext_vspace_below_ \__enumext_level: _skip } }
1250         }
1251         {
1252             \vspace { \skip_use:c { l__enumext_vspace_below_ \__enumext_level: _skip } }
1253         }
1254     }
1255 }

```

(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.

```

1256 \cs_new_protected:Nn \__enumext_vspace_above_v:
1257 {
1258   \skip_if_eq:nnF { \l__enumext_vspace_above_v_skip } { \c_zero_skip }
1259   {
1260     \bool_if:NTF \l__enumext_vspace_a_star_v_bool
1261     {
1262       \vspace*{ \l__enumext_vspace_above_v_skip }
1263     }
1264     { \vspace { \l__enumext_vspace_above_v_skip } }
1265   }
1266 }
```

(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.

```

1267 \cs_new_protected:Nn \__enumext_vspace_below_v:
1268 {
1269   \skip_if_eq:nnF { \l__enumext_vspace_below_v_skip } { \c_zero_skip }
1270   {
1271     \bool_if:NTF \l__enumext_vspace_b_star_v_bool
1272     {
1273       \vspace*{ \l__enumext_vspace_below_v_skip }
1274     }
1275     { \vspace { \l__enumext_vspace_below_v_skip } }
1276   }
1277 }
```

(End of definition for `__enumext_vspace_below_v:`.)

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

`__enumext_vspace_above_vii:`

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

`__enumext_vspace_above_viii:`

```

1278 \cs_new_protected:Nn \__enumext_vspace_above_vii:
1279 {
1280   \skip_if_eq:nnF { \l__enumext_vspace_above_vii_skip } { \c_zero_skip }
1281   {
1282     \bool_if:NTF \l__enumext_vspace_a_star_vii_bool
1283     {
1284       \vspace*{ \l__enumext_vspace_above_vii_skip }
1285     }
1286     { \vspace { \l__enumext_vspace_above_vii_skip } }
1287   }
1288 }
1289 \cs_new_protected:Nn \__enumext_vspace_above_viii:
1290 {
1291   \skip_if_eq:nnF { \l__enumext_vspace_above_viii_skip } { \c_zero_skip }
1292   {
1293     \bool_if:NTF \l__enumext_vspace_a_star_viii_bool
1294     {
1295       \vspace*{ \l__enumext_vspace_above_viii_skip }
1296     }
1297     { \vspace { \l__enumext_vspace_above_viii_skip } }
1298   }
1299 }
```

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

`__enumext_vspace_below_vii:`

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

`__enumext_vspace_below_viii:`

```

1300 \cs_new_protected:Nn \__enumext_vspace_below_vii:
1301 {
1302   \skip_if_eq:nnF { \l__enumext_vspace_below_vii_skip } { \c_zero_skip }
1303   {
1304     \bool_if:NTF \l__enumext_vspace_b_star_vii_bool
1305     {
1306       \vspace*{ \l__enumext_vspace_below_vii_skip }

```

```

1307     }
1308     { \vspace { \l__enumext_vspace_below_vii_skip } }
1309   }
1310 }
1311 \cs_new_protected:Nn \__enumext_vspace_below_viii:
1312 {
1313   \skip_if_eq:nnF { \l__enumext_vspace_below_viii_skip } { \c_zero_skip }
1314   {
1315     \bool_if:NTF \l__enumext_vspace_b_star_viii_bool
1316     {
1317       \vspace*{ \l__enumext_vspace_below_viii_skip }
1318     }
1319     { \vspace { \l__enumext_vspace_below_viii_skip } }
1320   }
1321 }

```

(End of definition for __enumext_vspace_below_vii: and __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.

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

```

resume
resume*
1322 \keys_define:nn { enumext / level-1 }
1323 {
1324   save-ans .code:n = \__enumext_storing_set:n {#1},
1325   save-ans .value_required:n = true,
1326   resume .code:n = \__enumext_resume_counter:,
1327   resume .value_forbidden:n = true,
1328   resume* .code:n = \__enumext_resume_counter_star:,
1329   resume* .value_forbidden:n = true,
1330 }
1331 \keys_define:nn { enumext / enumext* }
1332 {
1333   save-ans .code:n = \__enumext_storing_set:n {#1},
1334   save-ans .value_required:n = true,
1335   resume .code:n = \__enumext_resume_counter_vii:,
1336   resume .value_forbidden:n = true,
1337 }

```

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

`__enumext_storing_set:n`

The function `__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 `__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.

```

1338 \cs_new_protected:Npn \__enumext_storing_set:n #1
1339 {
1340   \tl_set:Nx \l__enumext_store_name_tl {#1}
1341   \bool_set_true:N \l__enumext_store_active_bool
1342   \int_if_exist:cF { g__enumext_resume_#1_int }
1343   {
1344     \int_new:c { g__enumext_resume_#1_int }
1345   }
1346   \bool_if:NT \l__enumext_check_ans_bool
1347   {
1348     \__enumext_check_ans_int:n {#1}
1349   }
1350 }

```

(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`.

```

1351 \cs_new_protected:Nn \__enumext_resume_counter:

```

```

1352 {
1353   \bool_if:NT \l__enumext_store_active_bool
1354   {
1355     \int_gset:Nn \g__enumext_resume_int
1356     {
1357       \int_use:c { g__enumext_resume_ \l__enumext_store_name_tl _int }
1358     }
1359   }
1360   \bool_set_true:N \l__enumext_resume_bool
1361 }
1362 \cs_new_protected:Nn \__enumext_resume_counter_vii:
1363 {
1364   \bool_if:NT \l__enumext_store_active_bool
1365   {
1366     \int_gset:Nn \g__enumext_resume_int
1367     {
1368       \int_use:c { g__enumext_resume_ \l__enumext_store_name_tl _int }
1369     }
1370   }
1371   \bool_set_true:N \l__enumext_resume_vii_bool
1372 }

```

(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 begins with `\item` or `\item*` and does NOT open a nested environment, each `\item` or `\item*` must contain a *single* execution of the `\anskey` command, i.e. the counter of the executions of the `\anskey` command must be equal to the counter associated with the sum of executions of `\item` and `\item*`.

If the line begins with `\item` or `\item*` and opens a nested environment each `\item` or `\item*` in the nested environment must have a *single* execution of the `\anskey` command and the counter associated to the sum of `\item` and `\item*` executions must increment by “one” to maintain equality.

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

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

The integer variable associated to the sum of each `\item` and `\item*` in the environment `\g__enumext_count_item_all_int` must match the integer variable `\g__enumext_count_item_ans_int` associated to the execution of the command `\anskey`. We analyze the cases:

- a) If the list only has one level the number of `\item` + `\item*` = `\anskey`
- b) If the list has *nested levels*, for each level of nesting we need to 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 no-store
1373 \cs_set_protected:Npn \__enumext_tmp:n #1
1374 {
1375   \keys_define:nn { enumext / #1 }
1376   {
1377     check-ans .bool_set:N = \l__enumext_check_ans_bool,
1378     check-ans .initial:n = false,
1379     no-store .code:n = {
1380       \bool_set_false:N \l__enumext_store_ans_bool
1381       \bool_set_false:N \l__enumext_check_ans_bool
1382     },

```

```

1383         no-store .value_forbidden:n = true,
1384     }
1385 }
1386 \clist_map_inline:nn
1387 {
1388     level-1, level-2, level-3, level-4, enumext*
1389 }
1390 { \__enumext_tmp:n {#1} }

```

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

`__enumext_check_ans_int:n`

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`

```

1391 \cs_new_protected:Npn \__enumext_check_ans_int:n #1
1392 {
1393     \int_if_exist:cF { g__enumext_count_#1_item_ans_int }
1394     { \int_new:c { g__enumext_count_#1_item_ans_int } }
1395     \int_if_exist:cF { g__enumext_count_#1_i_int }
1396     { \int_new:c { g__enumext_count_#1_i_int } }
1397     \int_if_exist:cF { g__enumext_count_#1_ii_int }
1398     { \int_new:c { g__enumext_count_#1_ii_int } }
1399     \int_if_exist:cF { g__enumext_count_#1_iii_int }
1400     { \int_new:c { g__enumext_count_#1_iii_int } }
1401     \int_if_exist:cF { g__enumext_count_#1_iv_int }
1402     { \int_new:c { g__enumext_count_#1_iv_int } }
1403     \int_if_exist:cF { g__enumext_count_#1_vii_int }
1404     { \int_new:c { g__enumext_count_#1_vii_int } }

```

We make `\g__enumext_count_item_all_int` equal to the integer variables `\g__enumext_count_⟨store name⟩_item_i_int` or `\g__enumext_count_⟨store name⟩_item_vii_int` that contains all the occurrences of `\item` and `\item*` in the different levels and we will make `\g__enumext_count_item_with_ans_int` equal to the integer variable handled by the `\anskey` command.

```

1405     \bool_lazy_all:nTF
1406     {
1407         { \g__enumext_starred_bool }
1408         { \int_compare_p:nNn { \l__enumext_level_int } = { \c_zero_int } }
1409     }
1410     {
1411         \int_gset_eq:Nc \g__enumext_count_item_all_int { g__enumext_count_#1_vii_int }
1412     }
1413     {
1414         \int_gset_eq:Nc \g__enumext_count_item_all_int { g__enumext_count_#1_i_int }
1415     }
1416     \int_gset_eq:Nc \g__enumext_count_item_i_int { g__enumext_count_#1_i_int }
1417     \int_gset_eq:Nc \g__enumext_count_item_ii_int { g__enumext_count_#1_ii_int }
1418     \int_gset_eq:Nc \g__enumext_count_item_iii_int { g__enumext_count_#1_iii_int }
1419     \int_gset_eq:Nc \g__enumext_count_item_iv_int { g__enumext_count_#1_iv_int }
1420     \int_gset_eq:Nc \g__enumext_count_item_vii_int { g__enumext_count_#1_vii_int }
1421     \int_gset_eq:Nc \g__enumext_count_item_with_ans_int { g__enumext_count_#1_item_ans_int }
1422 }

```

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

10.22.2 Set-up check answer mechanism

`__enumext_check_ans_count:`

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.

```

1423 \cs_new_protected:Npn \__enumext_check_ans_count:
1424 {
1425     \bool_if:NT \l__enumext_check_ans_bool
1426     {
1427         \bool_if:NTF \l__enumext_store_ans_bool
1428         {
1429             \int_gadd:cn { g__enumext_count_item_ \__enumext_level: _int }
1430             { \int_use:c { g__enumext_count_level_ \__enumext_level: _int } + 1 }
1431         }
1432         { \int_gincr:c { g__enumext_count_item_ \__enumext_level: _int } }
1433     }
1434 }

```

(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:`

```

1435 \cs_new_protected:Nn \__enumext_check_ans_active:
1436 {
1437   \int_set:Nn \l__enumext_compare_items_ans_int
1438   {
1439     \g__enumext_count_item_all_int - \g__enumext_count_item_ii_int
1440     - \g__enumext_count_item_iii_int - \g__enumext_count_item_iv_int
1441   }
1442   \int_compare:nNnTF
1443   { \l__enumext_compare_items_ans_int } = { \g__enumext_count_item_with_ans_int }
1444   {
1445     \msg_term:nnV { enumext } { items-same-answer } \g__enumext_store_name_tl
1446   }
1447   {
1448     \msg_warning:nnV { enumext } { item-different-answer } \g__enumext_store_name_tl
1449   }

```

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

```

1450   \__enumext_zero_count_level:
1451 }

1452 \cs_new_protected:Nn \__enumext_check_ans_active_vii:
1453 {
1454   \int_set:Nn \l__enumext_compare_items_ans_int
1455   {
1456     \g__enumext_count_item_all_int - \g__enumext_count_item_i_int
1457     - \g__enumext_count_item_ii_int - \g__enumext_count_item_iii_int
1458     - \g__enumext_count_item_iv_int
1459   }
1460   \int_compare:nNnTF
1461   { \l__enumext_compare_items_ans_int } = { \g__enumext_count_item_with_ans_int }
1462   {
1463     \msg_term:nnV { enumext } { items-same-answer } \g__enumext_store_name_tl
1464   }
1465   {
1466     \msg_warning:nnV { enumext } { item-different-answer } \g__enumext_store_name_tl
1467   }
1468   \__enumext_zero_count_level:
1469 }

```

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

10.23 Keys and functions associated with storage

`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*`.

```

1470 \cs_set_protected:Npn \__enumext_tmp:n #1
1471 {
1472   \keys_define:nn { enumext / #1 }
1473   {
1474     wrap-ans .cs_set_protected:Np = \__enumext_anskey_wrapper:n #1,
1475     wrap-ans .initial:n = \fbox{##1},
1476     wrap-ans .value_required:n = true,
1477     mark-ans .code:n = \tl_set:Nn \l__enumext_mark_answer_sym_tl {##1},
1478     mark-ans .initial:n = \textasteriskcentered,
1479     mark-ans .value_required:n = true,
1480     mark-pos .choice:,
1481     mark-pos / left .code:n = \str_set:Nn \l__enumext_mark_position_str { l },
1482     mark-pos / right .code:n = \str_set:Nn \l__enumext_mark_position_str { r },
1483     mark-pos .initial:n = right,
1484     mark-pos .value_required:n = true,
1485     show-ans .code:n = \bool_set_true:N \l__enumext_show_answer_bool
1486               \bool_set_false:N \l__enumext_show_position_bool,
1487     show-ans .value_forbidden:n = true,
1488     show-pos .code:n = \bool_set_true:N \l__enumext_show_position_bool
1489               \bool_set_false:N \l__enumext_show_answer_bool,
1490     show-pos .value_forbidden:n = true,
1491     mark-ref .code:n = \tl_set:Nn \l__enumext_mark_ref_sym_tl {##1},

```



```

1492     mark-ref .initial:n = \textasteriskcentered,
1493     mark-ref .value_required:n = true,
1494     store-ref .bool_set:N = \l__enumext_store_ref_key_bool,
1495     store-ref .initial:n = false,
1496   }
1497 }
1498 \clist_map_inline:nn { level-1, enumext* } { \l__enumext_tmp:n {#1} }

```

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

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

```

1499 \cs_set_protected:Npn \l__enumext_tmp:n #1
1500 {
1501   \keys_define:nn { enumext / #1 }
1502   {
1503     mark-pos .choice:,
1504     mark-pos / left .code:n = \str_set:Nn \l__enumext_mark_position_str { l },
1505     mark-pos / right .code:n = \str_set:Nn \l__enumext_mark_position_str { r },
1506     mark-pos .initial:n = right,
1507     mark-pos .value_required:n = true,
1508     show-ans .code:n = \bool_set_true:N \l__enumext_show_answer_bool
1509                     \bool_set_false:N \l__enumext_show_position_bool,
1510     show-ans .value_forbidden:n = true,
1511     show-pos .code:n = \bool_set_true:N \l__enumext_show_position_bool
1512                     \bool_set_false:N \l__enumext_show_answer_bool,
1513     show-pos .value_forbidden:n = true,
1514   }
1515 }
1516 \clist_map_inline:nn { keyans, keyans* } { \l__enumext_tmp:n {#1} }

```

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

columns* For the `enumext` and `enumext*` environments we will only add the keys `columns*` and `columns-sep*`.
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 `\l__enumext_store_level_open:` function used by the “storage system” to preserve the structure and then used by the `\printkeyans` command.

```

1517 \cs_set_protected:Npn \l__enumext_tmp:nn #1 #2
1518 {
1519   \keys_define:nn { enumext / #1 }
1520   {
1521     columns* .code:n = \bool_set_true:c { \l__enumext_store_columns_#2_bool }
1522                     \int_set:cn { \l__enumext_store_columns_#2_int } {##1}
1523                     \tl_put_right:ce { \l__enumext_store_opt_#2_tl }
1524                     {
1525                       columns = \exp_not:v { \l__enumext_store_columns_#2_int },
1526                     },,
1527     columns* .value_required:n = true,
1528     columns-sep* .code:n = \bool_set_true:c { \l__enumext_store_columns_sep_#2_bool }
1529                     \dim_set:cn { \l__enumext_store_columns_sep_#2_dim } {##1}
1530                     \tl_put_right:ce { \l__enumext_store_opt_#2_tl }
1531                     {
1532                       columns-sep = \exp_not:v { \l__enumext_store_columns_sep_#2_dim },
1533                     },
1534     columns-sep* .value_required:n = true,
1535   }
1536 }
1537 \clist_map_inline:nn
1538 {
1539   {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {enumext*}{vii}
1540 }
1541 { \l__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 $\langle prop list \rangle$ 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 $\langle prop list \rangle$ 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.

```
1542 \cs_new_protected:Npn \__enumext_store_addto_prop:n #1
1543 {
1544   \prop_if_exist:cF { g__enumext_ \l__enumext_store_name_tl _prop }
1545   { \prop_new:c { g__enumext_ \l__enumext_store_name_tl _prop } }
1546   \prop_gput:cen { g__enumext_ \l__enumext_store_name_tl _prop }
1547   {
1548     \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } +1 }
1549     } { #1 }
1550 }
1551 \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 $\langle sequence \rangle$ 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 $\langle sequence \rangle$ 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.

```
1552 \cs_new_protected:Npn \__enumext_store_addto_seq:n #1
1553 {
1554   \seq_if_exist:cF { g__enumext_ \l__enumext_store_name_tl _seq }
1555   { \seq_new:c { g__enumext_ \l__enumext_store_name_tl _seq } }
1556   \seq_gput_right:cn { g__enumext_ \l__enumext_store_name_tl _seq } { #1 }
1557 }
1558 \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 `0pt` and pass them as an optional argument to the environment stored in the sequence `enumext`.

```
1559 \cs_new_protected:Nn \__enumext_store_level_open:
1560 {
1561   \bool_if:NF \l__enumext_store_ans_bool
1562   {
1563     \tl_if_empty:cTF { l__enumext_store_opt_ \l__enumext_level: _tl }
1564     {
1565       \__enumext_store_addto_seq:n
1566       {
1567         \item \begin{enumext}
1568       }
1569     }
1570     {
1571       \tl_put_left:cn { l__enumext_store_opt_ \l__enumext_level: _tl }
1572       {
1573         \item \begin{enumext} [
1574       }
1575       \tl_put_right:cn { l__enumext_store_opt_ \l__enumext_level: _tl }
1576       {
1577         ]
1578       }
1579       \__enumext_store_addto_seq:v { l__enumext_store_opt_ \l__enumext_level: _tl }
1580     }
1581   }
1582 }
1583 \cs_new_protected:Nn \__enumext_store_level_close:
```

```

1584 {
1585   \bool_if:NF \l__enumext_store_ans_bool
1586   {
1587     \__enumext_store_addto_seq:n { \end{enumext} }
1588   }
1589 }

```

(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.

```

1590 \cs_new_protected:Nn \__enumext_store_level_open_vii:
1591 {
1592   \bool_if:NF \l__enumext_store_ans_bool
1593   {
1594     \tl_if_empty:NTF \l__enumext_store_opt_vii_tl
1595     {
1596       \__enumext_store_addto_seq:n
1597       {
1598         \item \mode_leave_vertical:
1599         \vspace { -\skip_eval:n { \baselineskip + \parsep } }
1600         \begin{enumext*}[before={\setlength{\topsep}{\opt}},]
1601       }
1602     }
1603     {
1604       \tl_put_left:Nn \l__enumext_store_opt_vii_tl
1605       {
1606         \item \mode_leave_vertical:
1607         \vspace { -\skip_eval:n { \baselineskip + \parsep } }
1608         \begin{enumext*}[before={\setlength{\topsep}{\opt}},
1609       ]
1610       \tl_put_right:Nn \l__enumext_store_opt_vii_tl
1611       {
1612         ]
1613       }
1614       \__enumext_store_addto_seq:V \l__enumext_store_opt_vii_tl
1615     }
1616   }
1617 }
1618 \cs_new_protected:Nn \__enumext_store_level_close_vii:
1619 {
1620   \bool_if:NF \l__enumext_store_ans_bool
1621   {
1622     \__enumext_store_addto_seq:n { \end{enumext*} }
1623   }
1624 }

```

(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`

```

1625 \cs_new_protected:Nn \__enumext_print_keyans_box:NN
1626 {
1627   \mode_leave_vertical:
1628   \skip_horizontal:n { -\dim_use:N #2 }
1629   \makebox[\opt][ r ]
1630   {
1631     \makebox[ \dim_use:N #1 ][ \l__enumext_mark_position_str ]
1632     {
1633       \tl_use:N \l__enumext_mark_answer_sym_tl
1634     }
1635   }
1636   \skip_horizontal:n { \dim_use:N #2 }

```

```

1637 }
1638 \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`.

```

1639 \keys_define:nn { enumext / anskey }
1640 {
1641   item-sym* .tl_set:N = \l__enumext_store_item_symbol_tl,
1642   item-sym* .value_required:n = true,
1643   item-pos* .dim_set:N = \l__enumext_store_item_symbol_sep_dim,
1644   item-pos* .value_required:n = true,
1645   store-brk .bool_set:N = \l__enumext_store_columns_break_bool,
1646   store-brk .default:n = true,
1647   store-brk .value_forbidden:n = true,
1648 }

```

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_with_ans_int`, then call internal function `__enumext_store_anskey_code:nnnn` will “*store content*” in the *(sequence)* and in the *(prop list)*.

```

1649 \NewDocumentCommand \anskey { d() s o +m }
1650 {
1651   \bool_if:NF \l__enumext_store_active_bool
1652   {
1653     \msg_error:nnnn { enumext } { anskey-wrong-place } { anskey } { enumext }
1654   }
1655   \int_compare:nNnT { \l__enumext_keyans_level_int } = { 1 }
1656   {
1657     \msg_error:nnnn { enumext } { command-wrong-place } { anskey } { keyans }
1658   }
1659   \int_compare:nNnT { \l__enumext_keyans_pic_level_int } = { 1 }
1660   {
1661     \msg_error:nnnn { enumext } { command-wrong-place } { anskey } { keyanspic }
1662   }
1663   \group_begin:
1664     \bool_if:NF \l__enumext_store_ans_bool
1665     {
1666       \bool_if:NT \l__enumext_check_ans_bool
1667       {
1668         \int_gincr:N \g__enumext_count_item_with_ans_int
1669       }
1670       \__enumext_store_anskey_code:nnnn {#1} {#2} {#3} {#4}
1671     }
1672   \group_end:
1673 }

```

(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 `__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.

```

1674 \cs_new_protected:Npn \__enumext_store_anskey_code:nnnn #1 #2 #3 #4
1675 {
1676   \__enumext_store_addto_prop:n {#4}
1677   \bool_if:NT \l__enumext_store_ref_key_bool
1678   {
1679     \__enumext_store_internal_ref:
1680   }
1681   \__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`.

```

1682 \tl_clear:N \l__enumext_store_anskey_arg_tl
1683 \tl_if_novalue:nF {#3}
1684 {
1685   \keys_set:nn { enumext / anskey } {#3}
1686 }
1687 \bool_lazy_and:nnT
1688 { \l__enumext_store_columns_break_bool }
1689 { \bool_not_p:n { \l__enumext_starred_bool } }
1690 {
1691   \tl_put_left:Nn \l__enumext_store_anskey_arg_tl { \columnbreak }
1692 }
1693 \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).

```

1694 \tl_if_novalue:nF {#1}
1695 {
1696   \int_set:Nn \l__enumext_store_columns_join_int {#1}
1697   \bool_if:NT \l__enumext_starred_bool
1698   {
1699     \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
1700     {
1701       ( \exp_not:V \l__enumext_store_columns_join_int )
1702     }
1703   }
1704 }

```

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`.

```

1705 \bool_if:nTF {#2}
1706 {
1707   \tl_put_right:Nn \l__enumext_store_anskey_arg_tl { * }
1708   \tl_if_empty:NF \l__enumext_store_item_symbol_tl
1709   {
1710     \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
1711     {
1712       [ \exp_not:V \l__enumext_store_item_symbol_tl ]
1713     }
1714   }
1715   \dim_compare:nT
1716   {
1717     \l__enumext_store_item_symbol_sep_dim != \c_zero_dim
1718   }
1719   {
1720     \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
1721     {
1722       [ \exp_not:V \l__enumext_store_item_symbol_sep_dim ]
1723     }
1724   }
1725   \tl_put_right:Nn \l__enumext_store_anskey_arg_tl {#4}
1726 }
1727 {
1728   \tl_put_right:Nn \l__enumext_store_anskey_arg_tl {#4}
1729 }

```

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*.

```

1730 \bool_lazy_and:nnT
1731 { \l__enumext_store_ref_key_bool }
1732 { \l__enumext_hyperref_bool }
1733 {
1734   \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
1735   {
1736     \hfill \exp_not:N \hyperlink { \exp_not:V \l__enumext_newlabel_arg_one_tl }
1737     { \exp_not:V \l__enumext_mark_ref_sym_tl }
1738   }

```

```

1739     }
1740     \__enumext_store_addto_seq:V \__enumext_store_anskey_arg_tl
1741 }

```

(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 __enumext_newlabel_arg_two_tl.

```

1742 \cs_new_protected:Nn \__enumext_store_internal_ref:
1743 {
1744   \cs_set_protected:Npn \__enumext_tmp:n ##1
1745   {
1746     \tl_set_eq:cc { \__enumext_label_copy_##1_tl } { \__enumext_label_##1_tl }
1747     \tl_reverse:c { \__enumext_label_copy_##1_tl }
1748     \tl_remove_once:cn { \__enumext_label_copy_##1_tl } { . }
1749     \tl_reverse:c { \__enumext_label_copy_##1_tl }
1750   }
1751   \clist_map_inline:nn { i, ii, iii, iv, vii } { \__enumext_tmp:n {##1} }
1752   \cs_set:Npn \__enumext_tmp:n ##1
1753   { . \tl_use:c { \__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.

```

1754   \bool_lazy_all:nT
1755   {
1756     { \g__enumext_starred_bool }
1757     { \int_compare_p:nNn { \__enumext_level_int } = { \c_zero_int } }
1758   }
1759   {
1760     \tl_put_right:Ne \__enumext_newlabel_arg_two_tl
1761     { \tl_use:N \__enumext_label_copy_vii_tl }
1762   }
1763   \bool_lazy_all:nT
1764   {
1765     { \l__enumext_standar_bool }
1766     { \g__enumext_starred_bool }
1767     { \int_compare_p:nNn { \__enumext_level_int } > { \c_zero_int } }
1768   }
1769   {
1770     \tl_put_right:Ne \__enumext_newlabel_arg_two_tl
1771     {
1772       \tl_use:N \__enumext_label_copy_vii_tl
1773       \int_step_function:nnN { 1 } { \__enumext_level_int } \__enumext_tmp:n
1774     }
1775   }

```

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.

```

1776   \bool_lazy_all:nT
1777   {
1778     { \l__enumext_standar_bool }
1779     { \int_compare_p:nNn { \__enumext_level_int } > { \c_zero_int } }
1780     { \int_compare_p:nNn { \__enumext_level_h_int } = { \c_zero_int } }
1781     { \bool_not_p:n { \l__enumext_starred_bool } }
1782   }
1783   {
1784     \tl_put_right:Ne \__enumext_newlabel_arg_two_tl
1785     {
1786       \tl_use:N \__enumext_label_copy_i_tl
1787       \int_step_function:nnN { 2 } { \__enumext_level_int } \__enumext_tmp:n
1788     }
1789   }
1790   \cs_set:Npn \__enumext_tmp:n ##1
1791   { \tl_use:c { \__enumext_label_copy_ \int_to_roman:n {##1} _tl } }
1792   \bool_lazy_all:nT
1793   {
1794     { \l__enumext_standar_bool }
1795     { \int_compare_p:nNn { \__enumext_level_int } > { \c_zero_int } }

```

```

1796     { \bool_not_p:n { \g__enumext_starred_bool } }
1797     { \int_compare_p:nNn { \l__enumext_level_h_int } > { \c_zero_int } }
1798   }
1799   {
1800     \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
1801     {
1802       \int_step_function:nnN { 1 } { \l__enumext_level_int } \__enumext_tmp:n
1803       . \tl_use:N \l__enumext_label_copy_vii_tl
1804     }
1805   }

```

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

```

1806     \tl_put_right:Ne \l__enumext_newlabel_arg_one_tl
1807     {
1808       \l__enumext_store_name_tl \c_colon_str
1809       \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } }
1810     }

```

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.

```

1811     \tl_put_right:Ne \l__enumext_store_write_aux_file_tl
1812     {
1813       \__enumext_newlabel:nn
1814       { \exp_not:V \l__enumext_newlabel_arg_one_tl }
1815       { \l__enumext_newlabel_arg_two_tl }
1816     }
1817     \l__enumext_store_write_aux_file_tl
1818   }

```

(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.

```

1819 \cs_new_protected:Npn \__enumext_store_anskey_show_wrap:n #1
1820 {
1821   \par
1822   \bool_if:NT \l__enumext_starred_bool
1823   {
1824     \cs_set:Nn \__enumext_level: { vii }
1825   }
1826   \__enumext_print_keyans_box:cc
1827   { \l__enumext_labelwidth_ \__enumext_level: _dim }
1828   { \l__enumext_labelsep_ \__enumext_level: _dim }
1829   \__enumext_anskey_wrapper:n { #1 }
1830 }

```

(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.

```

1831 \cs_new_protected:Npn \__enumext_store_anskey_show_left:n #1
1832 {
1833   \bool_if:NT \l__enumext_show_answer_bool
1834   {
1835     \__enumext_store_anskey_show_wrap:n { #1 }
1836   }
1837   \bool_if:NT \l__enumext_show_position_bool
1838   {
1839     \tl_set:Ne \l__enumext_mark_answer_sym_tl
1840     {
1841       \group_begin:
1842       \exp_not:N \normalfont
1843       \exp_not:N \footnotesize [ \int_eval:n
1844       {
1845         \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
1846       }
1847       ]
1848       \group_end:

```



```

1849     }
1850     \__enumext_store_anskey_show_wrap:n { #1 }
1851   }
1852 }

```

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

10.25 Common functions for `keyans`, `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`.

```

1853 \cs_new_protected:Npn \__enumext_keyans_addto_prop:n #1
1854 {
1855   \tl_clear:N \l__enumext_store_keyans_label_tl
1856   \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
1857   {
1858     \tl_put_right:Ne \l__enumext_store_keyans_label_tl { \l__enumext_label_vi_tl }
1859   }
1860   {
1861     \tl_put_right:Ne \l__enumext_store_keyans_label_tl { \l__enumext_label_v_tl }
1862   }
1863   \tl_if_novalue:nF { #1 }
1864   {
1865     \tl_put_right:Ne \l__enumext_store_keyans_label_tl { \c_space_tl #1 }
1866   }
1867   \__enumext_store_addto_prop:V \l__enumext_store_keyans_label_tl
1868 }

```

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

10.25.2 The store-ref key for `keyans`, `keyans*` and `keyanspic`

The internal “*label and ref*” system for the `keyans`, `keyans*` and `keyanspic` environments has slight differences with the one implemented for the `\anskey` command, basically because in this environments we are interested in the current *label*. The mechanism defined here will allow to execute `\ref{<store name : position>}` and will return 1. (A).

`__enumext_keyans_store_ref:`
`__enumext_keyans_store_ref_aux_i:`
`__enumext_keyans_store_ref_aux_ii:`

The function `__enumext_keyans_store_ref:` handles the internal “*label and ref*” system used by the `store-ref` key for `\item*` and `\anspic*` commands. First we will create copies of the current *labels* and remove the dots “.” from them, we do not want to get double dots in our references.

```

1869 \cs_new_protected:Nn \__enumext_keyans_store_ref:
1870 {
1871   \bool_if:NT \l__enumext_store_ref_key_bool
1872   {
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     \__enumext_keyans_store_ref_aux_i:
1882   }
1883 }

```

The auxiliary function `__enumext_keyans_store_ref_aux_i:` set the variable `\l__enumext_newlabel_arg_one_tl` which will contain `{<store name : position>}` analyzing whether the environment in which they are executed is `enumext*` or `enumext`.

```

1884 \cs_new_protected:Nn \__enumext_keyans_store_ref_aux_i:
1885 {
1886   \bool_if:NT \g__enumext_starred_bool
1887   {
1888     \tl_set_eq:NN \l__enumext_label_copy_i_tl \l__enumext_label_copy_vii_tl
1889   }
1890   \int_compare:nNnT { \l__enumext_keyans_pic_level_int } = { 1 }

```

```

1891     {
1892       \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
1893       { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_vi_tl }
1894     }
1895     \int_compare:nNt { \l__enumext_keyans_level_int } = { 1 }
1896     {
1897       \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
1898       { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_v_tl }
1899     }
1900     \int_compare:nNt { \l__enumext_keyans_level_h_int } = { 1 }
1901     {
1902       \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
1903       { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_viii_tl }
1904     }
1905     \tl_put_right:Ne \l__enumext_newlabel_arg_one_tl
1906     {
1907       \l__enumext_store_name_tl \c_colon_str
1908       \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } }
1909     }
1910     \__enumext_keyans_store_ref_aux_ii:
1911   }

```

Now auxiliary function `__enumext_keyans_store_ref_aux_ii:` save the result in the variable `\l__enumext_store_write_aux_file_tl` and finally we write in the `.aux` file.

```

1912 \cs_new_protected:Nn \__enumext_keyans_store_ref_aux_ii:
1913 {
1914   \tl_put_right:Ne \l__enumext_store_write_aux_file_tl
1915   {
1916     \__enumext_newlabel:nn
1917     { \exp_not:V \l__enumext_newlabel_arg_one_tl }
1918     { \l__enumext_newlabel_arg_two_tl }
1919   }
1920   \l__enumext_store_write_aux_file_tl
1921 }

```

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

10.25.3 Storing content in sequence

`__enumext_keyans_addto_seq:n`
`__enumext_keyans_addto_seq_link:`

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

```

1922 \cs_new_protected:Npn \__enumext_keyans_addto_seq:n #1
1923 {
1924   \tl_clear:N \l__enumext_store_keyans_label_tl
1925   \int_compare:nNtF { \l__enumext_keyans_pic_level_int } = { 1 }
1926   {
1927     \tl_put_right:Ne \l__enumext_store_keyans_label_tl { \item \l__enumext_label_vi_tl }
1928   }
1929   {
1930     \tl_put_right:Ne \l__enumext_store_keyans_label_tl { \item \l__enumext_label_v_tl }
1931   }
1932   \tl_if_novalue:nF { #1 }
1933   {
1934     \tl_put_right:Ne \l__enumext_store_keyans_label_tl { \c_space_tl #1 }
1935   }
1936   \__enumext_keyans_addto_seq_link:
1937 }

```

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. 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_with_ans_int` handled by the `check-ans` key.

```

1938 \cs_new_protected:Nn \__enumext_keyans_addto_seq_link:
1939 {
1940   \bool_lazy_and:nnT
1941   { \l__enumext_store_ref_key_bool }

```

```

1942 { \l__enumext_hyperref_bool }
1943 {
1944   \tl_put_right:Ne \l__enumext_store_keyans_label_tl
1945   {
1946     \hflll \exp_not:N \hyperlink
1947     {
1948       \exp_not:V \l__enumext_newlabel_arg_one_tl
1949     }
1950     { \exp_not:V \l__enumext_mark_ref_sym_tl }
1951   }
1952 }
1953 \__enumext_store_addto_seq:V \l__enumext_store_keyans_label_tl
1954 \tl_gset:NV \g__enumext_check_ans_item_tl \l__enumext_store_keyans_label_tl
1955 \bool_if:NT \l__enumext_check_ans_bool
1956 {
1957   \int_gincr:N \g__enumext_count_item_with_ans_int
1958 }
1959 }

```

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

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.

```

1960 \cs_new_protected:Npn \__enumext_keyans_check_ans:nn #1 #2
1961 {
1962   \tl_if_empty:NTF \g__enumext_check_ans_item_tl
1963   {
1964     \msg_warning:nnnn { enumext } { missing-starred }{ #1 }{ #2 }
1965   }
1966   { \tl_gclear:N \g__enumext_check_ans_item_tl }
1967 }

```

(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.

```

1968 \cs_new_protected:Npn \__enumext_keyans_show_left:n #1
1969 {
1970   \bool_if:NT \l__enumext_show_answer_bool
1971   {
1972     \tl_put_left:Nn \l__enumext_label_v_tl
1973     {
1974       \__enumext_print_keyans_box:NN
1975       \l__enumext_labelwidth_i_dim
1976       \l__enumext_labelsep_i_dim
1977     }
1978     \tl_if_novalue:nF { #1 }
1979     { \tl_put_right:Nn \l__enumext_label_v_tl { \c_space_tl [ #1 ] } }
1980   }
1981   \bool_if:NT \l__enumext_show_position_bool
1982   {
1983     \tl_set:Ne \l__enumext_mark_answer_sym_tl
1984     {
1985       \group_begin:
1986       \exp_not:N \normalfont
1987       \exp_not:N \footnotesize [ \int_eval:n
1988         {
1989           \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
1990           + \l__enumext_keyans_level_int
1991         }
1992       ]
1993       \group_end:
1994     }
1995     \tl_put_left:Nn \l__enumext_label_v_tl
1996     {

```

```

1997         \__enumext_print_keyans_box:NN
1998         \l__enumext_labelwidth_i_dim
1999         \l__enumext_labelsep_i_dim
2000     }
2001     \tl_if_novalue:nF { #1 }
2002     { \tl_put_right:Nn \l__enumext_label_v_tl { \c_space_tl [ #1 ] } }
2003 }
2004 }

```

(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*.
item-pos*
2005 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
2006 {
2007     \keys_define:nn { enumext / #1 }
2008     {
2009         item-sym* .tl_set:c = { \l__enumext_item_symbol_#2_tl },
2010         item-sym* .value_required:n = true,
2011         item-sym* .initial:n = { $\star$ },
2012         item-pos* .dim_set:c = { \l__enumext_item_symbol_sep_#2_dim },
2013         item-pos* .value_required:n = true,
2014     }
2015 }
2016 \clist_map_inline:nn
2017 {
2018     {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {enumext*}{vii}
2019 }
2020 { \__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 Clea F. Rees (@cfr) in [footnotes in boxes compatible with hyperref](#).

```

2021 \cs_new_protected:Nn \__enumext_footnotetext:nn
2022 {
2023     \footnotetext[#1]{#2}
2024 }
2025 \cs_new_protected:Nn \__enumext_renew_footnote:
2026 {
2027     \seq_gclear:N \g__enumext_footnote_arg_seq
2028     \seq_gclear:N \g__enumext_footnote_int_seq
2029     \RenewDocumentCommand \footnote { o +m }
2030     {
2031         \tl_if_novalue:nTF {##1}
2032         {
2033             \stepcounter{footnote}
2034             \int_gset_eq:Nc \g__enumext_footnote_int { c@footnote }
2035         }
2036         {
2037             \int_gset:Nn \g__enumext_footnote_int { ##1 }
2038         }
2039         \footnotemark [ \g__enumext_footnote_int ]
2040         \seq_gput_right:Nn \g__enumext_footnote_arg_seq { ##2 }
2041         \seq_gput_right:NV \g__enumext_footnote_int_seq \g__enumext_footnote_int
2042     }
2043 }
2044 \cs_new_protected:Nn \__enumext_print_footnote:
2045 {
2046     \seq_if_empty:NF \g__enumext_footnote_int_seq
2047     {
2048         \seq_map_pairwise_function:NNN
2049         \g__enumext_footnote_int_seq
2050         \g__enumext_footnote_arg_seq
2051         \__enumext_footnotetext:nn

```

```

2052     }
2053 }

```

(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).

```

2054 \cs_new_protected:Npn \__enumext_default_item:n #1
2055 {
2056   \tl_if_novalue:nTF {#1}
2057   {
2058     \bool_if:NT \l__enumext_check_ans_bool
2059     {
2060       \int_gincr:N \g__enumext_count_item_all_int
2061       \int_gincr:c { g__enumext_count_level_ \__enumext_level: _int }
2062     }
2063     \bool_set_true:c { l__enumext_wrap_label_ \__enumext_level: _bool }
2064     \__enumext_item_std:w \tl_use:c { l__enumext_fake_item_indent_ \__enumext_level: _tl }
2065   }
2066   {
2067     \bool_set_eq:cc
2068     { l__enumext_wrap_label_ \__enumext_level: _bool }
2069     { l__enumext_wrap_label_opt_ \__enumext_level: _bool }
2070     \__enumext_item_std:w [#1] \tl_use:c { l__enumext_fake_item_indent_ \__enumext_level: _tl }
2071   }
2072 }

```

(End of definition for `__enumext_default_item:n`)

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

#1: `\l__enumext_item_symbol_X_tl`

#2: `\l__enumext_item_symbol_sep_X_dim`

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

Then we will see the state of the variable `\l__enumext_check_ans_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).

```

2073 \cs_new_protected:Npn \__enumext_starred_item:nn #1 #2
2074 {
2075   \tl_if_novalue:nF {#1}
2076   {
2077     \tl_set:cn { l__enumext_item_symbol_ \__enumext_level: _tl } {#1}
2078   }
2079   \tl_gset_eq:Nc \g__enumext_item_symbol_tl { l__enumext_item_symbol_ \__enumext_level: _tl }
2080   \tl_if_novalue:nTF {#2}
2081   {

```

```

2082     \dim_set_eq:cc
2083     { \__enumext_item_symbol_sep_ \__enumext_level: _dim }
2084     { \__enumext_labelsep_ \__enumext_level: _dim }
2085   }
2086   {
2087     \dim_set:cn { \__enumext_item_symbol_sep_ \__enumext_level: _dim } {#2}
2088   }
2089   \bool_if:NT \__enumext_check_ans_bool
2090   {
2091     \int_gincr:N \__enumext_count_item_all_int
2092     \int_gincr:c { \__enumext_count_level_ \__enumext_level: _int }
2093   }
2094   \bool_set_true:c { \__enumext_wrap_label_ \__enumext_level: _bool }
2095   \__enumext_item_std:w \tl_use:c { \__enumext_fake_item_indent_ \__enumext_level: _tl }
2096 }

```

(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).

```

2097 \cs_new_protected:Nn \__enumext_redefine_item:
2098 {
2099   \RenewDocumentCommand \item { s o o }
2100   {
2101     \bool_if:nTF {##1}
2102     {
2103       \__enumext_starred_item:nn {##2} {##3}
2104     }
2105     { \__enumext_default_item:n {##2} }
2106   }
2107 }

```

(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 $\langle label \rangle$ 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.

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

```

2108 \cs_new_protected:Npn \__enumext_keyans_default_item:n #1
2109 {
2110   \tl_if_novalue:nTF { #1 }
2111   {
2112     \bool_set_true:N \__enumext_wrap_label_v_bool
2113     \__enumext_item_std:w \tl_use:N \__enumext_fake_item_indent_v_tl
2114   }
2115   {
2116     \bool_set_eq:NN \__enumext_wrap_label_v_bool \__enumext_wrap_label_opt_v_bool
2117     \__enumext_item_std:w [#1] \tl_use:N \__enumext_fake_item_indent_v_tl
2118   }
2119 }

```

(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 $\langle label \rangle$, 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 $\langle label \rangle$.

```

2120 \cs_new_protected:Npn \__enumext_keyans_starred_item:n #1
2121 {
2122   \tl_set_eq:NN \__enumext_keyans_tmpa_tl \__enumext_label_v_tl
2123   \__enumext_keyans_show_left:n { #1 }
2124   \bool_set_true:N \__enumext_wrap_label_v_bool
2125   \__enumext_item_std:w \tl_use:N \__enumext_fake_item_indent_v_tl

```

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

```

2126 \tl_set_eq:NN \l__enumext_label_v_tl \l__enumext_keyans_tmpa_tl
2127 \__enumext_keyans_addto_prop:n { #1 }
2128 \__enumext_keyans_store_ref:
2129 \__enumext_keyans_addto_seq:n { #1 }
2130 }

```

(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).

```

2131 \cs_new_protected:Nn \__enumext_keyans_redefine_item:
2132 {
2133   \RenewDocumentCommand \item { s o }
2134   {
2135     \bool_if:nTF {##1}
2136     {
2137       \peek_remove_spaces:n
2138       {
2139         \__enumext_keyans_starred_item:n {##2}
2140       }
2141     }
2142     {
2143       \__enumext_keyans_default_item:n {##2}
2144     }
2145   }
2146 }

```

(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.

```

2147 \cs_new_protected:Nn \__enumext_item_starred:
2148 {
2149   \tl_if_empty:cF { \__enumext_item_symbol_ \__enumext_level: _tl }
2150   {
2151     \mode_leave_vertical:
2152     \skip_horizontal:n { -\dim_use:c { \__enumext_item_symbol_sep_ \__enumext_level: _dim } }
2153     \makebox[ 0pt ][ r ]{ \tl_use:N \g__enumext_item_symbol_tl }
2154     \skip_horizontal:n { \dim_use:c { \__enumext_item_symbol_sep_ \__enumext_level: _dim } }
2155   }
2156 }

```

(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).

```

2157 \cs_new_protected:Nn \__enumext_make_label:
2158 {
2159   \RenewDocumentCommand \makeLabel { m }
2160   {
2161     \tl_use:c { \__enumext_label_fill_left_ \__enumext_level: _tl }
2162     \tl_use:c { \__enumext_label_font_style_ \__enumext_level: _tl }
2163     \bool_if:cTF { \__enumext_wrap_label_ \__enumext_level: _bool }
2164     {
2165       \__enumext_item_starred:
2166       \use:c { __enumext_wrapper_label_ \__enumext_level: :n } { ##1 }

```



```
2167     }
2168     { ##1 }
2169     \tl_use:c { l__enumext_label_fill_right_ \__enumext_level: _tl }
2170     \tl_gclear:N \g__enumext_item_symbol_tl
2171   }
2172 }
```

(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).

```
2173 \cs_new_protected:Nn \__enumext_keyans_make_label:
2174 {
2175   \RenewDocumentCommand \makeLabel { m }
2176   {
2177     \tl_use:N \l__enumext_label_fill_left_v_tl
2178     \tl_use:N \l__enumext_label_font_style_v_tl
2179     \bool_if:NTF \l__enumext_wrap_label_v_bool
2180     {
2181       \__enumext_wrapper_label_v:n { ##1 }
2182     }
2183     { ##1 }
2184     \tl_use:N \l__enumext_label_fill_right_v_tl
2185   }
2186 }
```

(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.

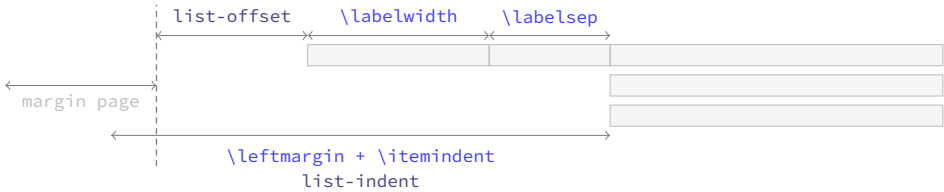


Figure 9: Representation of standard horizontal lengths in list environment.

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.

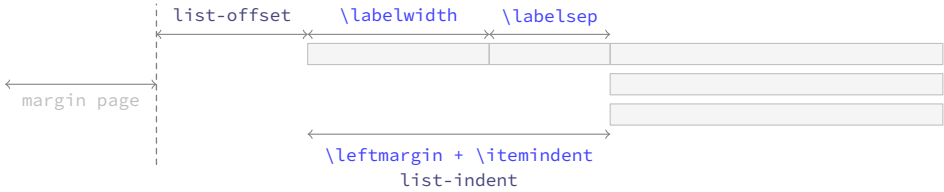


Figure 10: Representation of horizontal lengths concept in list in enumext.

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

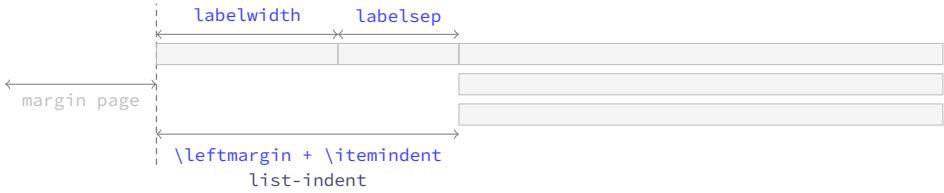


Figure 11: Default horizontal lengths in enumext.

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

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

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

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

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

```

2187 \cs_new_protected:Npn \__enumext_calc_hspace:NNNNNNN #1 #2 #3 #4 #5 #6 #7
2188 {
2189   \dim_compare:nNnT { #1 } < { \c_zero_dim }
2190   {
2191     \msg_warning:nnnV { enumext } { width-non-positive } { labelwidth } { #1 }
2192     \dim_set:Nn #1 { \dim_abs:n { #1 } }
2193   }
2194   \dim_compare:nNnT { #2 } < { \c_zero_dim }
2195   {
2196     \msg_warning:nnnV { enumext } { width-negative } { labelsep } { #2 }
2197     \dim_set:Nn #2 { \dim_abs:n { #2 } }
2198   }

```

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

```

2199   \bool_if:nF #7 { \dim_set:Nn #4 { #1 + #2 } }

```

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

```

2200   \dim_compare:nNnTF { #4 } < { \c_zero_dim }
2201   {
2202     \dim_set:Nn #6 { #1 + #2 - #4 }
2203     \dim_set:Nn #5 { #1 + #2 + #3 - #6 }
2204   }
2205   {
2206     \dim_compare:nNnT { #4 } = { #1 + #2 }
2207     { \dim_set:Nn #6 { \c_zero_dim } }
2208     \dim_compare:nNnT { #4 } < { #1 + #2 }
2209     { \dim_set:Nn #6 { #1 + #2 - #4 } }
2210     \dim_compare:nNnT { #4 } > { #1 + #2 }
2211     {
2212       \dim_set:Nn #6 { -#1 - #2 + #4 }
2213       \dim_set:Nn #6 { #6*-1 }
2214     }
2215     \dim_set:Nn #5 { #1 + #2 + #3 - #6 }
2216   }
2217 }
2218 \cs_generate_variant:Nn \__enumext_calc_hspace:NNNNNNN { cccccc }

```

(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.

```

\__enumext_list_arg_two_i:
\__enumext_list_arg_two_ii:
\__enumext_list_arg_two_iii:
\__enumext_list_arg_two_iv:
\__enumext_list_arg_two_v:
2219 \cs_set_protected:Npn \__enumext_tmp:n #1
2220 {
2221   \cs_new_protected:cpn { __enumext_list_arg_two_#1: }
2222   {
2223     \__enumext_calc_hspace:ccccc
2224     { \__enumext_labelwidth_#1_dim } { \__enumext_labelsep_#1_dim }
2225     { \__enumext_listoffset_#1_dim } { \__enumext_leftmargin_tmp_#1_dim }
2226     { \__enumext_leftmargin_#1_dim } { \__enumext_itemindent_#1_dim }
2227     { \__enumext_leftmargin_tmp_#1_bool }
2228     \clist_map_inline:nn
2229     { labelsep, labelwidth, itemindent, leftmargin, rightmargin, listparindent }
2230     { \dim_set_eq:cc {####1} { \__enumext_####1_#1_dim } }
2231     \clist_map_inline:nn { topsep, parsep, partopsep, itemsep }
2232     { \skip_set_eq:cc {####1} { \__enumext_####1_#1_skip } }
2233     \usecounter { enumX#1 }
2234     \bool_lazy_and:nnTF
2235     { \str_if_eq_p:nn {#1} { i } } {

```

```

2236         { \bool_if_p:N \__enumext_resume_bool }
2237         { \setcounter { enumXi } { \int_eval:n { \g__enumext_resume_int } } }
2238         {
2239             \setcounter { enumX#1 }
2240             { \int_eval:n { \int_use:c { \__enumext_start_#1_int } - 1 } }
2241         }
2242     \str_if_eq:nnTF {#1} { v }
2243     {
2244         \__enumext_keyans_redefine_item:
2245         \__enumext_keyans_make_label:
2246         \__enumext_keyans_fake_item:
2247         \bool_if:cT { \__enumext_show_length_#1_bool }
2248         {
2249             \msg_term:nnnn { enumext } { list-lengths-not-nested } { v } { keyans }
2250         }
2251     }
2252     {
2253         \__enumext_redefine_item:
2254         \__enumext_make_label:
2255         \__enumext_use_key_ref:
2256         \__enumext_fake_item:
2257         \bool_if:cT { \__enumext_show_length_#1_bool }
2258         {
2259             \msg_term:nnne { enumext } { list-lengths } {#1} { \int_use:N \__enumext_level_i
2260         }
2261     }
2262 }
2263 }
2264 \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.

```

2265 \cs_set_protected:Npn \__enumext_tmp:n #1
2266 {
2267     \cs_new_protected:cpn { __enumext_list_arg_two_#1: }
2268     {
2269         \__enumext_calc_hspace:cccccc
2270         { \__enumext_labelwidth_#1_dim } { \__enumext_labelsep_#1_dim }
2271         { \__enumext_listoffset_#1_dim } { \__enumext_leftmargin_tmp_#1_dim }
2272         { \__enumext_leftmargin_#1_dim } { \__enumext_itemindent_#1_dim }
2273         { \__enumext_leftmargin_tmp_#1_bool }
2274     \clist_map_inline:nn
2275     { labelsep, labelwidth, itemindent, leftmargin, rightmargin, listparindent }
2276     { \dim_set_eq:cc {####1} { \__enumext_####1_#1_dim } }
2277     \clist_map_inline:nn { topsep, parsep, partopsep, itemsep }
2278     { \skip_set_eq:cc {####1} { \__enumext_####1_#1_skip } }
2279     \skip_set_eq:Nc \parsep { \__enumext_itemsep_#1_skip }
2280     \skip_zero:N \partopsep
2281     \usecounter { enumX#1 }
2282     \bool_lazy_and:nnTF
2283     { \str_if_eq_p:nn {#1} { vii } } { \bool_if_p:N \__enumext_resume_vii_bool }
2284     { \setcounter { enumXvii } { \int_eval:n { \g__enumext_resume_vii_int } } }
2285     {
2286         \setcounter { enumX#1 }
2287         { \int_eval:n { \int_use:c { \__enumext_start_#1_int } - 1 } }
2288     }
2289     \__enumext_use_key_ref_h:
2290     \str_if_eq:nnTF {#1} { vii }
2291     {
2292         \__enumext_fake_item_vii:
2293         \bool_if:cT { \__enumext_show_length_vii_bool }
2294         { \msg_term:nnnn { enumext } { list-lengths-not-nested } { vii } { enumext* } }
2295     }
2296     {
2297         \__enumext_fake_item_viii:
2298         \bool_if:cT { \__enumext_show_length_#1_bool }
2299         { \msg_term:nnnn { enumext } { list-lengths-not-nested } { #1 } { keyans* } }

```

```

2300     }
2301   }
2302 }
2303 \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.

```

2304 \NewDocumentEnvironment{enumext}{ 0{} }
2305 {
2306   \__enumext_safe_exec:
2307   \__enumext_parse_keys:n {#1}
2308   \__enumext_before_list:
2309   \__enumext_start_store_level:
2310   \__enumext_start_list:nn
2311   { \tl_use:c { \__enumext_label_ \__enumext_level: _tl } }
2312   {
2313     \use:c { __enumext_list_arg_two_ \__enumext_level: : }
2314     \__enumext_before_keys_exec:
2315   }
2316   \__enumext_after_args_exec:
2317 }
2318 {
2319   \__enumext_stop_list:
2320   \__enumext_stop_store_level:
2321   \__enumext_after_list:
2322 }

```

(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 and set the state of the booleans `vars \l__enumext_standar_bool` and `\g__enumext_standar_bool` to “true”, the latter only if the environment is NOT nested in the `enumext*` environment.

```

2323 \cs_new_protected:Nn \__enumext_safe_exec:
2324 {
2325   \int_incr:N \l__enumext_level_int
2326   \int_compare:nNnT { \l__enumext_level_int } > { 4 }
2327   { \msg_fatal:nn { enumext } { list-too-deep } }
2328   \bool_set_true:N \l__enumext_standar_bool
2329   \bool_lazy_all:nT
2330   {
2331     { \bool_not_p:n { \l__enumext_starred_bool } }
2332     { \int_compare_p:nNn { \l__enumext_level_h_int } = { \c_zero_int } }
2333   }
2334   {
2335     \bool_gset_true:N \g__enumext_standar_bool
2336   }
2337 }

```

(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.

```

2338 \cs_new_protected:Npn \__enumext_parse_keys:n #1
2339 {
2340   \exp_args:Ne \keys_set:nn
2341   { enumext / level-\int_use:N \l__enumext_level_int } {#1}
2342   \bool_if:NT \l__enumext_store_active_bool
2343   {
2344     \__enumext_parse_store_keys:n {#1}
2345   }
2346 }

```

(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:`.

```

2347 \cs_new_protected:Npn \__enumext_parse_store_keys:n #1
2348 {
2349   \bool_if:cF { \l__enumext_store_columns_ \__enumext_level: _bool }
2350   {
2351     \regex_match:nnT { \b columns\b } {#1}
2352     {
2353       \int_set_eq:cc
2354       { \l__enumext_store_columns_ \__enumext_level: _int }
2355       { \l__enumext_columns_ \__enumext_level: _int }
2356       \tl_put_right:ce { \l__enumext_store_opt_ \__enumext_level: _tl }
2357       {
2358         columns = \exp_not:v { \l__enumext_store_columns_ \__enumext_level: _int },
2359       }
2360     }
2361   }
2362   \bool_if:cF { \l__enumext_store_columns_sep_ \__enumext_level: _bool }
2363   {
2364     \regex_match:nnT { \b columns-sep\b } {#1}
2365     {
2366       \dim_set_eq:cc
2367       { \l__enumext_store_columns_sep_ \__enumext_level: _dim }
2368       { \l__enumext_columns_sep_ \__enumext_level: _dim }
2369       \tl_put_right:ce { \l__enumext_store_opt_ \__enumext_level: _tl }
2370       {
2371         columns-sep = \exp_not:v { \l__enumext_store_columns_sep_ \__enumext_level: _dim }
2372       }
2373     }
2374   }
2375 }

```

(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.

`__enumext_stop_store_level:`

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

```

2376 \cs_new_protected:Nn \__enumext_start_store_level:
2377 {
2378   %\bool_lazy_and:nnT
2379   %{ \l__enumext_store_active_bool }
2380   %{ \bool_not_p:n { \l__enumext_keyans_env_bool } }
2381   %{
2382     %\int_compare:nNnT { \l__enumext_level_h_int } = { 1 }
2383     %{
2384       %\bool_set_true:c { \l__enumext_store_upper_level_ \__enumext_level: _bool }
2385       %\__enumext_store_level_open:
2386       %}
2387     %\int_compare:nNnT { \l__enumext_level_int } > { 1 }
2388     %{
2389       \bool_set_true:c { \l__enumext_store_upper_level_ \__enumext_level: _bool }
2390       \__enumext_store_level_open:
2391       %}
2392     %}
2393   }
2394 \cs_new_protected:Nn \__enumext_stop_store_level:
2395 {
2396   \bool_if:cT { \l__enumext_store_upper_level_ \__enumext_level: _bool }
2397   {
2398     \__enumext_store_level_close:
2399   }
2400 }

```

(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.

```

2401 \cs_new_protected:Nn \__enumext_before_list:
2402 {
2403   \__enumext_vspace_above:
2404   \__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.

```

2405   \__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.

```

2406   \dim_compare:nNnT
2407   { \dim_use:c { \l__enumext_minipage_right_ \__enumext_level: _dim } } > { \c_zero_dim }
2408   {
2409     \dim_set:cn { \l__enumext_minipage_left_ \__enumext_level: _dim }
2410     {
2411       \linewidth
2412       - \dim_use:c { \l__enumext_minipage_right_ \__enumext_level: _dim }
2413       - \dim_use:c { \l__enumext_minipage_hsep_ \__enumext_level: _dim }
2414     }

```

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*.

```

2415     \bool_set_true:c { \l__enumext_minipage_active_ \__enumext_level: _bool }
2416     \int_gincr:N \g__enumext_minipage_stat_int
2417     \__enumext_mini_addvspace:
2418     \nointerlineskip\noindent
2419     \begin{__enumext_mini_env*}
2420     { \dim_use:c { \l__enumext_minipage_left_ \__enumext_level: _dim } }
2421   }
2422   \__enumext_multicols_start:
2423 }

```

(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.

```

2424 \cs_new_protected:Nn \__enumext_multicols_start:
2425 {
2426   \int_compare:nNnT
2427   { \int_use:c { \l__enumext_columns_ \__enumext_level: _int } } > { 1 }
2428   {
2429     \dim_compare:nNnT
2430     { \dim_use:c { \l__enumext_columns_sep_ \__enumext_level: _dim } } = { \c_zero_dim }
2431     {
2432       \dim_set:cn { \l__enumext_columns_sep_ \__enumext_level: _dim }
2433       {
2434         ( \dim_use:c { \l__enumext_labelwidth_ \__enumext_level: _dim }
2435         + \dim_use:c { \l__enumext_labelsep_ \__enumext_level: _dim }
2436         ) / \int_use:c { \l__enumext_columns_ \__enumext_level: _int }
2437         - \dim_use:c { \l__enumext_listoffset_ \__enumext_level: _dim }
2438       }
2439     }
2440     \dim_set_eq:Nc \columnsep { \l__enumext_columns_sep_ \__enumext_level: _dim }
2441     \skip_zero:N \multicolsep
2442     \int_compare:nNnT { \l__enumext_level_int } > { 1 }
2443     {
2444       \dim_zero:N \columnseprule
2445     }

```

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.

```

2446         \bool_if:cF { \__enumext_minipage_active_ \__enumext_level: _bool }
2447         {
2448             \__enumext_multi_addvspace:
2449         }
2450     \raggedcolumns
2451     \begin{multicols}{\int_use:c { \__enumext_columns_ \__enumext_level: _int } }
2452 }
2453 }

```

(End of definition for `__enumext_multicols_start:`)

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

```

2454 \cs_new_protected:Nn \__enumext_multicols_stop:
2455 {
2456     \int_compare:nNt
2457     { \int_use:c { \__enumext_columns_ \__enumext_level: _int } } > { 1 }
2458     {
2459         \end{multicols}
2460         \bool_if:cF { \__enumext_minipage_active_ \__enumext_level: _bool }
2461         {
2462             \par\addvspace{ \skip_use:c { \__enumext_multicols_below_ \__enumext_level: _skip } }
2463         }
2464     }

```

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.

```

2465     \bool_lazy_and:nnT
2466     { \__enumext_check_ans_bool }
2467     { \bool_not_p:n { \g__enumext_starred_bool } }
2468     {
2469         \bool_gset_true:N \g__enumext_check_ans_show_bool
2470         \tl_gset:NV \g__enumext_store_name_tl \__enumext_store_name_tl
2471     }
2472 }

```

(End of definition for `__enumext_multicols_stop:`)

`__enumext_after_list:` The function `__enumext_after_list:` will check the state of the boolean variable `__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* `__enumext_minipage_after_skip`, otherwise we will close the `multicols` environment.

```

2473 \cs_new_protected:Nn \__enumext_after_list:
2474 {
2475     \bool_if:cTF { \__enumext_minipage_active_ \__enumext_level: _bool }
2476     {
2477         \int_compare:nNt { \g__enumext_minipage_stat_int } = { 1 }
2478         {
2479             \msg_warning:nn { enumext } { missing-miniright }
2480             \miniright
2481         }
2482         \int_gzero:N \g__enumext_minipage_stat_int
2483         \end{__enumext_mini_env*}
2484         \par\addvspace { \__enumext_minipage_after_skip }
2485     }
2486     { \__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.

```

2487     \__enumext_after_stop_list:
2488     \__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.

```

2489 \bool_set_false:N \l__enumext_standar_bool
2490 \int_gset_eq:NN \g__enumext_resume_int \value{enumXi}
2491 \int_if_exist:cT { g__enumext_resume_ \l__enumext_store_name_tl _int }
2492 {
2493   \int_gset_eq:cN
2494   { g__enumext_resume_ \l__enumext_store_name_tl _int }
2495   { \value{enumXi} }
2496 }
2497 }

```

(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`.

```

2498 \__enumext_after_env:nn {enumext}
2499 {
2500   \bool_if:NT \g__enumext_check_ans_show_bool
2501   {
2502     \int_compare:nNnT { \l__enumext_level_int } = { 0 }
2503     {
2504       \__enumext_check_ans_active:
2505     }
2506   }
2507   \bool_gset_false:N \g__enumext_check_ans_show_bool
2508   \tl_gclear:N \g__enumext_store_name_tl
2509 }

```

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.

```

2510 \NewDocumentEnvironment{keyans}{ 0{ } }
2511 {
2512   \__enumext_keyans_safe_exec:
2513   \__enumext_keyans_parse_keys:n {#1}
2514   \__enumext_before_list_v:
2515   \__enumext_start_list:nn
2516   { \tl_use:N \l__enumext_label_v_tl }
2517   {
2518     \__enumext_list_arg_two_v:
2519     \__enumext_before_keys_exec_v:
2520   }
2521   \__enumext_after_args_exec_v:
2522 }
2523 {
2524   \__enumext_keyans_check_ans:nn { item }{ keyans }
2525   \__enumext_stop_list:
2526   \__enumext_after_list_v:
2527 }

```

(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.

```

2528 \cs_new_protected:Nn \__enumext_keyans_safe_exec:
2529 {
2530   \bool_if:NF \l__enumext_store_active_bool
2531   {
2532     \msg_error:nnnn { enumext } { wrong-place }{ keyans }{ save-ans }
2533   }
2534   \int_incr:N \l__enumext_keyans_level_int
2535   \bool_set_true:N \l__enumext_keyans_env_bool
2536   % Set false for interfering with enumext nested in keyans (yes, its possible and crayze)
2537   \bool_set_false:N \l__enumext_store_active_bool
2538   \int_compare:nNnT { \l__enumext_keyans_level_int } > { 1 }

```

```

2539     {
2540       \msg_error:nn { enumext } { keyans-nested }
2541     }
2542     \int_compare:nNt { \l__enumext_level_int } > { 1 }
2543     {
2544       \msg_error:nn { enumext } { keyans-wrong-level }
2545     }
2546   }

```

(End of definition for `__enumext_keyans_safe_exec:`)

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

```

2547 \cs_new_protected:Npn \__enumext_keyans_parse_keys:n #1
2548 {
2549   \keys_set:nn { enumext / keyans } {#1}
2550 }

```

(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.

```

2551 \cs_new_protected:Nn \__enumext_before_list_v:
2552 {
2553   \__enumext_vspace_above_v:
2554   \__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.

```

2555   \dim_compare:nNt { \l__enumext_minipage_right_v_dim } > { \c_zero_dim }
2556   {
2557     \dim_set:Nn \l__enumext_minipage_left_v_dim
2558     {
2559       \linewidth - \l__enumext_minipage_right_v_dim - \l__enumext_minipage_hsep_v_dim
2560     }

```

The boolean variable `\l__enumext_minipage_active_v_bool` will be activated and the integer variable `\g__enumext_minipage_stat_int` used by the `\mini-right` 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*.

```

2561     \bool_set_true:N \l__enumext_minipage_active_v_bool
2562     \int_gincr:N \g__enumext_minipage_stat_int
2563     \__enumext_keyans_mini_addvspace:
2564     \nointerlineskip\noindent
2565     \begin{__enumext_mini_env*}{ \l__enumext_minipage_left_v_dim }
2566   }

```

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

```

2567   \__enumext_keyans_multicols_start:
2568 }

```

(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.

```

2569 \cs_new_protected:Nn \__enumext_keyans_multicols_start:
2570 {
2571   \int_compare:nNt { \l__enumext_columns_v_int } > { 1 }
2572   {

```

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

```

2573     \dim_compare:nNt { \l__enumext_columns_sep_v_dim } = { \c_zero_dim }
2574     {
2575       \dim_set:Nn \l__enumext_columns_sep_v_dim
2576       {
2577         (
2578           \l__enumext_labelwidth_v_dim + \l__enumext_labelsep_v_dim

```

```

2579         ) / \l__enumext_columns_v_int
2580         - \l__enumext_listoffset_v_dim
2581     }
2582 }
2583 \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).

```

2584 \skip_zero:N \multicolsep
2585 \dim_zero:N \columnseprule

```

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

```

2586 \bool_if:NF \l__enumext_minipage_active_v_bool
2587 {
2588     \__enumext_keyans_multi_advspace:
2589 }
2590 \raggedcolumns
2591 \begin{multicols}{\l__enumext_columns_v_int }
2592 }
2593 }

```

(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.

```

2594 \cs_new_protected:Nn \__enumext_keyans_multicols_stop:
2595 {
2596     \int_compare:nNnT { \l__enumext_columns_v_int } > { 1 }
2597     {
2598         \end{multicols}
2599         \bool_if:NF \l__enumext_minipage_active_v_bool
2600         {
2601             \par\addvspace{ \l__enumext_multicols_below_v_skip }
2602         }
2603     }
2604 }

```

(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.

```

2605 \cs_new_protected:Nn \__enumext_after_list_v:
2606 {
2607     \bool_if:NTF \l__enumext_minipage_active_v_bool
2608     {
2609         \int_compare:nNnT { \g__enumext_minipage_stat_int } = { 1 }
2610         {
2611             \msg_warning:nn { enumext } { missing-miniright }
2612             \miniright
2613         }
2614         \int_gzero:N \g__enumext_minipage_stat_int
2615         \end{\__enumext_mini-env*}
2616         \par\addvspace{ \l__enumext_minipage_after_skip }
2617     }
2618     { \__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.

```

2619 \bool_set_false:N \l__enumext_keyans_env_bool
2620 \__enumext_after_stop_list_v:
2621 \__enumext_vspace_below_v:
2622 }

```

(End of definition for `__enumext_after_list_v:`)

10.34 The environment key `anspic` and `\anspic`

The `keyanspic` environment is a list-based environment that uses the same configuration for “*spacing*” and $\langle label \rangle$ 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 $\langle label \rangle$ underneath, adjusting widths according to the options passed to the environment.

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

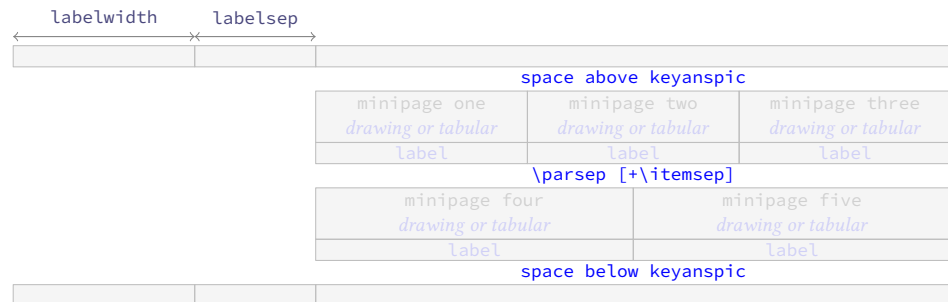


Figure 12: Representation of the `keyanspic` spacing in `enumext`.

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

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 `\sequence` and `\prop list` defined by `save-ans` key. This command is used as a replacement for `\item` in the `keyanspic` environment.

```
2623 \NewDocumentCommand \anspic { s o +m }
2624 {
```

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

```

2625 \bool_if:NF \l__enumext_store_active_bool
2626 {
2627     \msg_error:nnnn { enumext } { wrong-place }{ keyanspic }{ save-ans }
2628 }
2629 \int_compare:nNnT { \l__enumext_level_int } > { 1 }
2630 {
2631     \msg_error:nn { enumext } { keyanspic-wrong-level }
2632 }
2633 \int_compare:nNnT { \l__enumext_keyans_level_int } = { 1 }
2634 {
2635     \msg_error:nnnn { enumext } { command-wrong-place }{ ansPIC }{ keyans }
2636 }

```

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

```

2637 \seq_put_right:Nn \l__enumext_keyans_pic_body_seq
2638 {
2639   \__enumext_keyans_anspic_code:nnn { #1 } { #2 } { #3 }
2640 }
2641 }

```

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

```
\_enumext_keyans_anspic_code:nnn
```

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

```

2642 \cs_new_protected:Nn \__enumext_keyans_anspic_code:nno
2643 {
2644     \stepcounter { enumXvi }
2645     #3 \\\
2646     \bool_if:nT { #1 }
2647     {
2648         \__enumext_keyans_addto_prop:n { #2 }
2649         \__enumext_keyans_store_ref:
2650         \__enumext_keyans_addto_seq:n { #2 }
2651         \bool_lazy_or:nNT
2652             { \l_enumext_show_answer_bool }
2653             { \l_enumext_show_position_bool }

```

```

2654         {
2655             \tl_set_eq:NN \l__enumext_label_v_tl \l__enumext_label_vi_tl
2656             \__enumext_keyans_show_left:n { #2 }
2657             \tl_set_eq:NN \l__enumext_label_vi_tl \l__enumext_label_v_tl
2658         }
2659     }
2660     \tl_use:N \l__enumext_label_font_style_v_tl
2661     \__enumext_wrapper_label_v:n { \l__enumext_label_vi_tl }
2662 }

```

(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.

```

2663 \NewDocumentEnvironment{keyanspic}{o}
2664 {
2665     \__enumext_keyans_pic_safe_exec:
2666     \__enumext_start_list:nn
2667     { }
2668     {
2669         \__enumext_keyans_pic_arg_two:
2670     }

```

We apply the “adjusted” vertical spacing above the environment

```

2671     \vspace { \l__enumext_keyans_pic_above_skip }
2672 }

```

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.

```

2673 {
2674     \tl_if_novalue:nTF { #1 }
2675     {
2676         \__enumext_keyans_pic_do:e { \seq_count:N \l__enumext_keyans_pic_body_seq }
2677     }
2678     { \__enumext_keyans_pic_do:n { #1 } }
2679     \__enumext_stop_list:
2680     \__enumext_keyans_check_ans:nn { anspic } { keyanspic }
2681     \setcounter { enumXvi } { 0 }
2682     \vspace { \l__enumext_topsep_v_skip }
2683     %\bool_set_false:N \l__enumext_store_active_bool
2684 }

```

(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.

```

2685 \cs_new_protected:Nn \__enumext_keyans_pic_safe_exec:
2686 {
2687     \int_incr:N \l__enumext_keyans_pic_level_int
2688     \int_compare:nNnT { \l__enumext_keyans_pic_level_int } > { 1 }
2689     {
2690         \msg_error:nn { enumext } { keyanspic-nested }
2691     }
2692 }

```

(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**.

```

2693 \cs_new_protected:Npn \__enumext_keyans_pic_skip_abs:N #1
2694 {
2695     \dim_compare:nNnT { #1 } < { 0pt }
2696     { \skip_set:Nn #1 { -#1 } }
2697 }

```

(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.

```
2698 \cs_new_protected:Nn \__enumext_keyans_pic_arg_two:
2699 {
```

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.

```
2700     \bool_set_false:N \l__enumext_leftmargin_tmp_v_bool
2701     \__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.

```
2702     \skip_add:Nn \parsep { \itemsep }
2703     \dim_add:Nn \leftmargin { -\labelwidth - \labelsep }
2704     \dim_zero:N \labelwidth
2705     \dim_zero:N \listparindent
2706     \dim_zero:N \labelsep
2707     \skip_zero:N \partopsep
2708     \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 L^AT_EX when not using the `\item` command.

```
2709     \__enumext_keyans_pic_skip_abs:N \parsep
2710     \skip_set:Nn \l__enumext_keyans_pic_above_skip
2711     {
2712         \box_dp:N \strutbox
2713         + \l__enumext_topsep_v_skip
2714         - \parsep
2715     }
2716     \__enumext_item_std:w \scan_stop:
2717 }
```

(End of definition for `__enumext_keyans_pic_arg_two:.`)

`__enumext_keyans_pic_do:n` 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`.

```
2718 \cs_new_protected:Nn \__enumext_keyans_pic_do:n
2719 {
2720     \clist_map_function:nN { #1 } \__enumext_keyans_pic_row:n
2721 }
2722 \cs_generate_variant:Nn \__enumext_keyans_pic_do:n { e }
```

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

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

```
2723 \cs_new_protected:Nn \__enumext_keyans_pic_row:n
2724 {
2725     \dim_set:Nn \l__enumext_keyans_pic_width_dim { \linewidth / #1 }
2726     \int_set:Nn \l__enumext_keyans_pic_above_int { \l__enumext_keyans_pic_below_int }
2727     \int_set:Nn \l__enumext_keyans_pic_below_int { \l__enumext_keyans_pic_above_int + #1 }
2728     \int_step_inline:nnn
2729     { \l__enumext_keyans_pic_above_int + 1 }
2730     { \l__enumext_keyans_pic_below_int }
2731     {
2732         \__enumext_minipage:w [ b ]{ \l__enumext_keyans_pic_width_dim }
2733         \centering
2734         \seq_item:Nn \l__enumext_keyans_pic_body_seq { ##1 }
2735         \__enumext_endminipage:
2736     }
2737     \par
2738 }
```

(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 L^AT_EX 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 a 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

We set the default value for the width of the box containing the content of the items and create `\itemwidth` in a public form.

```

2739 \cs_new_protected:Nn \__enumext_starred_columns_set_vii:
2740 {
2741   \dim_compare:nNt { \l__enumext_columns_sep_vii_dim } = { \c_zero_dim }
2742   {
2743     \dim_set:Nn \l__enumext_columns_sep_vii_dim
2744     {
2745       ( \l__enumext_labelwidth_vii_dim + \l__enumext_labelsep_vii_dim )
2746       / \l__enumext_columns_vii_int
2747     }
2748   }
2749   \int_set:Nn \l__enumext_tmpa_vii_int { \l__enumext_columns_vii_int - \c_one_int }
2750   \dim_set:Nn \l__enumext_item_width_vii_dim
2751   {
2752     ( \linewidth - \l__enumext_columns_sep_vii_dim * \l__enumext_tmpa_vii_int )
2753     / \l__enumext_columns_vii_int - \l__enumext_labelwidth_vii_dim
2754     - \l__enumext_labelsep_vii_dim
2755   }
2756   \dim_zero_new:N \itemwidth
2757 }

```

(End of definition for `__enumext_starred_columns_set_vii:`.)

The function `__enumext_starred_joined_item_vii:n` will set the *width* of the box in which the content passed to `\item(\langle number \rangle)` will be stored together with the value of `\itemwidth`.

```

2758 \cs_new_protected:Npn \__enumext_starred_joined_item_vii:n #1
2759 {
2760   \int_set:Nn \l__enumext_joined_item_vii_int {#1}
2761   \int_compare:nNt { \l__enumext_joined_item_vii_int } > { \l__enumext_columns_vii_int }
2762   {
2763     \msg_warning:nnee { enumext } { item-joined }
2764     { \int_use:N \l__enumext_joined_item_vii_int }
2765     { \int_use:N \l__enumext_columns_vii_int }
2766     \int_set:Nn \l__enumext_joined_item_vii_int
2767     {
2768       \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + \c_one_int
2769     }
2770   }
2771   \int_compare:nNt
2772   { \l__enumext_joined_item_vii_int }
2773   >
2774   { \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + \c_one_int }
2775   {
2776     \msg_warning:nnee { enumext } { item-joined-columns }
2777     { \int_use:N \l__enumext_joined_item_vii_int }
2778     {
2779       \int_eval:n
2780       { \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + \c_one_int }
2781     }
2782     \int_set:Nn \l__enumext_joined_item_vii_int
2783     {
2784       \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + \c_one_int
2785     }
2786   }
}

```


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

```

2787 \int_compare:nNnTF { \l__enumext_joined_item_vii_int } > { \c_one_int }
2788 {
2789   \int_set_eq:NN \l__enumext_joined_item_aux_vii_int \l__enumext_joined_item_vii_int
2790   \int_decr:N \l__enumext_joined_item_aux_vii_int
2791   \int_add:Nn \l__enumext_item_column_pos_vii_int { \l__enumext_joined_item_aux_vii_int }
2792   \int_gadd:Nn \g__enumext_item_count_all_vii_int { \l__enumext_joined_item_aux_vii_int }
2793   \dim_set:Nn \l__enumext_joined_width_vii_dim
2794   {
2795     \l__enumext_item_width_vii_dim * \l__enumext_joined_item_vii_int
2796     + ( \l__enumext_labelwidth_vii_dim + \l__enumext_labelsep_vii_dim
2797       + \l__enumext_columns_sep_vii_dim
2798       ) * \l__enumext_joined_item_aux_vii_int
2799   }
2800   \dim_set_eq:NN \itemwidth \l__enumext_joined_width_vii_dim
2801 }
2802 {
2803   \dim_set_eq:NN \l__enumext_joined_width_vii_dim \l__enumext_item_width_vii_dim
2804   \dim_set_eq:NN \itemwidth \l__enumext_item_width_vii_dim
2805 }
2806 }

```

(End of definition for \l__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`.

```

2807 \cs_new_protected:Nn \__enumext_start_mini_vii:
2808 {
2809   \dim_compare:nNnT { \l__enumext_minipage_right_vii_dim } > { \c_zero_dim }
2810   {
2811     \dim_set:Nn \l__enumext_minipage_left_vii_dim
2812     {
2813       \linewidth
2814       - \l__enumext_minipage_right_vii_dim
2815       - \l__enumext_minipage_hsep_vii_dim
2816     }
2817     \bool_set_true:N \l__enumext_minipage_active_vii_bool
2818     \dim_gset_eq:NN
2819       \g__enumext_minipage_right_vii_dim
2820       \l__enumext_minipage_right_vii_dim
2821     \__enumext_mini_addvspace_vii:
2822     \nointerlineskip\noindent
2823     \begin{__enumext_mini_env*}{ \l__enumext_minipage_left_vii_dim }
2824   }
2825 }

```

(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*”.

```

2826 \cs_new_protected:Nn \__enumext_stop_mini_vii:
2827 {
2828   \bool_if:NT \l__enumext_minipage_active_vii_bool
2829   {
2830     \end{__enumext_mini_env*}
2831     \hfill
2832     \bool_gset_true:N \g__enumext_minipage_active_vii_bool
2833   }
2834 }

```

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*”.

```

2835 \__enumext_after_env:nn {enumext*}
2836 {
2837   \bool_if:NT \g__enumext_minipage_active_vii_bool
2838   {

```

```

2839 \begin{__enumext_mini_env*}{ \g__enumext_minipage_right_vii_dim }
2840 \par\addvspace { \g__enumext_minipage_right_skip }
2841 \bool_if:NF \g__enumext_minipage_center_vii_bool
2842 {
2843   \centering
2844 }
2845 \tl_use:N \g__enumext_miniright_code_vii_tl % the code
2846 \end{__enumext_mini_env*}
2847 \par\addvspace{ \g__enumext_minipage_after_skip }
2848 }
2849 \bool_gset_false:N \g__enumext_minipage_active_vii_bool
2850 \bool_gset_true:N \g__enumext_minipage_center_vii_bool
2851 \tl_gclear:N \g__enumext_miniright_code_vii_tl
2852 \dim_gzero:N \g__enumext_minipage_right_vii_dim
2853 }

```

(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.

```

2854 \NewDocumentEnvironment{enumext*}{ o }
2855 {
2856   \__enumext_safe_exec_vii:
2857   \__enumext_parse_keys_vii:n {#1}
2858   \__enumext_before_list_vii:
2859   \__enumext_start_store_level_vii:
2860   \__enumext_start_list:nn { }
2861   {
2862     \__enumext_list_arg_two_vii:
2863     \__enumext_before_keys_exec_vii:
2864   }
2865   \__enumext_starred_columns_set_vii:
2866   \item[] \scan_stop:
2867   \cs_set_eq:NN \__enumext_stop_item_tmp_vii: \noindent
2868   \cs_set_eq:NN \item \__enumext_start_item_tmp_vii:
2869 }
2870 {
2871   \__enumext_stop_item_tmp_vii:
2872   \__enumext_remove_extra_parsep_vii:
2873   \__enumext_stop_list:
2874   \__enumext_stop_store_level_vii:
2875   \__enumext_after_list_vii:
2876 }

```

(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.

```

2877 \cs_new_protected:Nn \__enumext_safe_exec_vii:
2878 {
2879   \int_incr:N \l__enumext_level_h_int
2880   \int_compare:nNnT { \l__enumext_level_h_int } > { 1 }
2881   {
2882     \msg_error:nn { enumext } { nested }
2883   }
2884   \bool_set_true:N \l__enumext_starred_bool
2885   \bool_lazy_all:nT
2886   {
2887     { \bool_not_p:n { \l__enumext_standar_bool } }
2888     { \int_compare_p:nNn { \l__enumext_level_int } = { \c_zero_int } }
2889   }
2890   {
2891     \bool_gset_true:N \g__enumext_starred_bool
2892   }
2893 }

```

(End of definition for __enumext_safe_exec_vii:.)

`__enumext_parse_keys_vii:n`

Parse [*key = val*] for `enumext*`. If the variable `__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.

```

2894 \cs_new_protected:Npn \__enumext_parse_keys_vii:n #1
2895 {
2896     \tl_if_novalue:nF {#1}
2897     {
2898         \keys_set:nn { enumext / enumext* } {#1}
2899         \bool_if:NT \__enumext_store_active_bool
2900         {
2901             \__enumext_parse_store_keys_vii:n {#1}
2902         }
2903     }
2904 }

```

(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 `__enumext_store_opt_vii_tl` which is used by the function `__enumext_store_level_open_vii:.`

```

2905 \cs_new_protected:Npn \__enumext_parse_store_keys_vii:n #1
2906 {
2907     \bool_if:NF \__enumext_store_columns_vii_bool
2908     {
2909         \regex_match:nnT { \b columns\b } {#1}
2910         {
2911             \int_set_eq:NN
2912             \__enumext_store_columns_vii_int
2913             \__enumext_columns_vii_int
2914             \tl_put_right:Ne \__enumext_store_opt_vii_tl
2915             {
2916                 columns = \exp_not:V \__enumext_store_columns_vii_int ,
2917             }
2918         }
2919     }
2920     \bool_if:NF \__enumext_store_columns_sep_vii_bool
2921     {
2922         \regex_match:nnT { \b columns-sep\b } {#1}
2923         {
2924             \dim_set_eq:NN
2925             \__enumext_store_columns_sep_vii_dim
2926             \__enumext_columns_sep_vii_dim
2927             \tl_put_right:Ne \__enumext_store_opt_vii_tl
2928             {
2929                 columns-sep = \exp_not:V \__enumext_store_columns_sep_vii_dim,
2930             }
2931         }
2932     }
2933 }

```

(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`.

```

2934 \cs_new_protected:Nn \__enumext_before_list_vii:
2935 {
2936     \__enumext_vspace_above_vii:
2937     \__enumext_before_args_exec_vii:
2938     \__enumext_start_mini_vii:
2939 }

```

(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.

```

2940 \cs_new_protected:Nn \__enumext_after_list_vii:
2941 {
2942   \__enumext_stop_mini_vii:
2943   \__enumext_after_stop_list_vii:
2944   \__enumext_vspace_below_vii:
2945   \int_gset_eq:NN \g__enumext_resume_vii_int \value{enumXvii}
2946   \int_if_exist:cT { \g__enumext_resume_ \l__enumext_store_name_tl _int }
2947   {
2948     \int_gset_eq:cN
2949     { \g__enumext_resume_ \l__enumext_store_name_tl _int }
2950     { \value{enumXvii} }
2951   }
2952   \bool_lazy_and:nnT { \g__enumext_starred_bool } { \l__enumext_check_ans_bool }
2953   {
2954     \bool_gset_true:N \g__enumext_check_ans_show_h_bool
2955     \tl_gset:NV \g__enumext_store_name_tl \l__enumext_store_name_tl
2956   }
2957   \bool_gset_false:N \g__enumext_starred_bool
2958   \bool_set_false:N \l__enumext_starred_bool
2959 }

```

(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`.

```

2960 \cs_new_protected:Nn \__enumext_start_store_level_vii:
2961 {
2962   \bool_if:NT \l__enumext_store_active_bool
2963   {
2964     \int_compare:nNnT { \l__enumext_level_int } > { \c_zero_int }
2965     {
2966       \__enumext_store_level_open_vii:
2967     }
2968   }
2969 }
2970 \cs_new_protected:Nn \__enumext_stop_store_level_vii:
2971 {
2972   \bool_if:NT \l__enumext_store_active_bool
2973   {
2974     \int_compare:nNnT { \l__enumext_level_int } > { \c_zero_int }
2975     {
2976       \__enumext_store_level_close_vii:
2977     }
2978   }
2979 }

```

(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`.

```

2980 \cs_new_protected_nopar:Nn \__enumext_start_item_tmp_vii:
2981 {
2982   \__enumext_stop_item_tmp_vii:
2983   \int_incr:N \l__enumext_item_column_pos_vii_int
2984   \int_gincr:N \g__enumext_item_count_all_vii_int
2985   \__enumext_item_peek_args_vii:
2986 }

```

(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).

```
2987 \cs_new_protected:Nn \__enumext_item_peek_args_vii:
2988 {
2989   \peek_meaning:NTF (
2990     { \__enumext_joined_item_vii:w }
2991     { \__enumext_joined_item_vii:w (1) }
2992   }
```

(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.

```
2993 \cs_new_protected:Npn \__enumext_joined_item_vii:w (#1)
2994 {
2995   \__enumext_starred_joined_item_vii:n {#1}
2996   \peek_meaning_remove:NTF *
2997   { \__enumext_starred_item_vii:w }
2998   { \__enumext_standard_item_vii:w }
2999 }
```

(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].

```
3000 \cs_new_protected:Npn \__enumext_standard_item_vii:w
3001 {
3002   \bool_set_false:N \l__enumext_item_starred_vii_bool
3003   \peek_meaning:NTF [
3004     {
3005       \bool_set_eq:NN
3006         \l__enumext_wrap_label_vii_bool
3007         \l__enumext_wrap_label_opt_vii_bool
3008       \__enumext_start_item_vii:w
3009     }
3010     {
3011       \bool_set_true:N \l__enumext_wrap_label_vii_bool
3012       \legacy_if_set_true:n { @noitemarg }
3013       \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
3014     }
3015 }
```

(End of definition for __enumext_standard_item_vii:w.)

__enumext_starred_item_vii:w
 __enumext_starred_item_vii_aux_i:w
 __enumext_starred_item_vii_aux_ii:w
 __enumext_starred_item_vii_aux_iii:w

The function __enumext_starred_item_vii:w together with the specified auxiliary functions `aux_i:w`, `aux_ii:w`, and `aux_iii:w` execute \item*, \item*[⟨*symbol*⟩] and \item*[⟨*symbol*⟩][⟨*offset*⟩].

```
3016 \cs_new_protected:Npn \__enumext_starred_item_vii:w
3017 {
3018   \bool_set_true:N \l__enumext_item_starred_vii_bool
3019   \bool_set_true:N \l__enumext_wrap_label_vii_bool
3020   \peek_meaning:NTF [
3021     { \__enumext_starred_item_vii_aux_i:w }
3022     { \__enumext_starred_item_vii_aux_ii:w }
3023   }
3024   \cs_new_protected:Npn \__enumext_starred_item_vii_aux_i:w [#1]
3025   {
3026     \tl_gset:Nn \g__enumext_item_symbol_aux_vii_tl {#1}
3027     \__enumext_starred_item_vii_aux_ii:w
3028   }
3029   \cs_new_protected:Npn \__enumext_starred_item_vii_aux_ii:w
3030   {
3031     \peek_meaning:NTF [
```

```

3032 { \__enumext_starred_item_vii_aux_iii:w }
3033 {
3034   \dim_set_eq:NN
3035     \l__enumext_item_symbol_sep_vii_dim
3036     \l__enumext_labelsep_vii_dim
3037   \legacy_if_set_true:n { @noitemarg }
3038   \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
3039 }
3040 }
3041 \cs_new_protected:Npn \__enumext_starred_item_vii_aux_iii:w [#1]
3042 {
3043   \dim_set:Nn \l__enumext_item_symbol_sep_vii_dim {#1}
3044   \legacy_if_set_true:n { @noitemarg }
3045   \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
3046 }

```

(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.

```

3047 \cs_new_protected_nopar:Npn \__enumext_start_item_vii:w [#1]
3048 {
3049   \cs_set_eq:NN \__enumext_stop_item_tmp_vii: \__enumext_stop_item_vii:
3050   \legacy_if:nT { @noitemarg }
3051   {
3052     \legacy_if_set_false:n { @noitemarg }
3053     \legacy_if:nT { @nmbrrlist }
3054     {
3055       \bool_if:NT \l__enumext_hyperref_bool
3056       {
3057         \legacy_if_set_true:n { @hyper@item }
3058       }
3059       \refstepcounter{enumXvii}
3060       % code for check-ans
3061       \bool_if:NT \l__enumext_check_ans_bool
3062       {
3063         % If true |no-store| key => nested in |enumext|
3064         \bool_if:NTF \l__enumext_store_ans_bool
3065         {
3066           \int_gadd:cn { g__enumext_count_item_ \__enumext_level: _int }
3067           { \int_use:c { g__enumext_count_level_ \__enumext_level: _int } + 1 }
3068         }
3069         {
3070           \int_gincr:N \g__enumext_count_item_all_int
3071           \int_gincr:N \g__enumext_count_level_vii_int
3072         }
3073       }
3074     }
3075   }

```

Here we start capturing \item and its contents into a group using the plain form of the `lrbox` environment. If the state of the variable \l__enumext_footnotes_key_bool is false, we will redefine the command \footnote, followed by printing the ⟨symbol⟩ defined for \item* if it is present and open a new group inside which we execute font key next to \item and the keys wrap-label, wrap-label*, align, close the group and execute the key labelsep and then the key first. Finally we open the minipage environment and execute the listparindent key which will be equal to \parindent, the parsep key which will be equal to \parskip and the itemindent key.

```

3076 \group_begin:
3077   \lrbox{ \l__enumext_item_text_vii_box }
3078   \bool_if:NF \l__enumext_footnotes_key_bool
3079   {
3080     \__enumext_renew_footnote:
3081   }

```

```

3082     \bool_if:NT \l__enumext_item_starred_vii_bool
3083     {
3084         \tl_if_blank:VT \g__enumext_item_symbol_aux_vii_tl
3085         {
3086             \tl_gset_eq:NN
3087             \g__enumext_item_symbol_aux_vii_tl \l__enumext_item_symbol_vii_tl
3088         }
3089         \mode_leave_vertical:
3090         \skip_horizontal:n { -\l__enumext_item_symbol_sep_vii_dim }
3091         \makebox[ 0pt ][ r ]{ \g__enumext_item_symbol_aux_vii_tl }
3092         \skip_horizontal:N \l__enumext_item_symbol_sep_vii_dim
3093         \tl_gclear:N \g__enumext_item_symbol_aux_vii_tl
3094     }
3095     \group_begin:
3096     \tl_use:N \l__enumext_label_font_style_vii_tl
3097     \bool_if:NTF \l__enumext_wrap_label_vii_bool
3098     {
3099         \makebox[ \l__enumext_labelwidth_vii_dim ][ \l__enumext_align_label_vii_str ]
3100         { \__enumext_wrapper_label_vii:n {#1} }
3101     }
3102     {
3103         \makebox[ \l__enumext_labelwidth_vii_dim ][ \l__enumext_align_label_vii_str ]{ #1 }
3104     }
3105     \group_end:
3106     \skip_horizontal:N \l__enumext_labelsep_vii_dim
3107     \tl_use:N \l__enumext_after_list_args_vii_tl
3108     \__enumext_minipage:w [ t ]{ \l__enumext_joined_width_vii_dim }
3109     \skip_set_eq:NN \parindent \l__enumext_listparindent_vii_dim
3110     \skip_set_eq:NN \parskip \l__enumext_parsep_vii_skip
3111     \tl_use:N \l__enumext_fake_item_indent_vii_tl
3112 }

```

(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.

```

3113 \cs_new_protected_nopar:Nn \__enumext_stop_item_vii:
3114 {
3115     \__enumext_endminipage:
3116     \endlrbox
3117     \group_end:
3118     \box_set_wd:Nn \l__enumext_item_text_vii_box
3119     {
3120         \l__enumext_joined_width_vii_dim
3121         + \l__enumext_labelwidth_vii_dim
3122         + \l__enumext_labelsep_vii_dim
3123     }
3124     \int_set:Nn \hbadness { 10000 }
3125     \box_use:N \l__enumext_item_text_vii_box
3126     \bool_if:NF \l__enumext_footnotes_key_bool
3127     {
3128         \__enumext_print_footnote:
3129     }
3130     \int_compare:nNnTF { \l__enumext_item_column_pos_vii_int } = { \l__enumext_columns_vii_int }
3131     {
3132         \par\noindent
3133         \int_zero:N \l__enumext_item_column_pos_vii_int
3134     }
3135     { \hspace{ \l__enumext_columns_sep_vii_dim } }
3136 }

```

(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.

```

3137 \cs_new_protected:Nn \__enumext_remove_extra_parsep_vii:
3138 {
3139     \int_compare:nNnT
3140     {
3141         \int_mod:nn { \g__enumext_item_count_all_vii_int } { \l__enumext_columns_vii_int }

```



```

3142     }
3143     =
3144     { \c_zero_int }
3145     {
3146         \par
3147         \vspace{ -\l__enumext_itemsep_vii_skip }
3148         \int_gzero:N \g__enumext_item_count_all_vii_int
3149     }
3150 }

```

(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`.

```

3151 \__enumext_after_env:nn {enumext*}
3152 {
3153     \bool_if:NT \g__enumext_check_ans_show_h_bool
3154     {
3155         \int_compare:nNnT { \l__enumext_level_int } = { 0 }
3156         {
3157             \__enumext_check_ans_active_vii:
3158         }
3159     }
3160     \bool_gset_false:N \g__enumext_check_ans_show_h_bool
3161     \tl_gclear:N \g__enumext_store_name_tl
3162 }

```

10.36 The `keyans*` environment

10.36.1 Functions for item box width

`__enumext_starred_columns_set_viii:`

We set the default value for the width of the box containing the content of the items and create `\itemwidth` in a public form.

```

3163 \cs_new_protected:Nn \__enumext_starred_columns_set_viii:
3164 {
3165     \dim_compare:nNnT { \l__enumext_columns_sep_viii_dim } = { \c_zero_dim }
3166     {
3167         \dim_set:Nn \l__enumext_columns_sep_viii_dim
3168         {
3169             ( \l__enumext_labelwidth_viii_dim + \l__enumext_labelsep_viii_dim )
3170             / \l__enumext_columns_viii_int
3171         }
3172     }
3173     \int_set:Nn \l__enumext_tmpa_viii_int { \l__enumext_columns_viii_int - \c_one_int }
3174     \dim_set:Nn \l__enumext_item_width_viii_dim
3175     {
3176         ( \linewidth - \l__enumext_columns_sep_viii_dim * \l__enumext_tmpa_viii_int )
3177         / \l__enumext_columns_viii_int - \l__enumext_labelwidth_viii_dim
3178         - \l__enumext_labelsep_viii_dim
3179     }
3180     \dim_zero_new:N \itemwidth
3181 }

```

(End of definition for `__enumext_starred_columns_set_viii:.`)

`__enumext_starred_joined_item_viii:n`

The function `__enumext_starred_joined_item_viii:n` will set the *width* of the box in which the content passed to `\item(<number>)` will be stored together with the value of `\itemwidth`.

```

3182 \cs_new_protected:Npn \__enumext_starred_joined_item_viii:n #1
3183 {
3184     \int_set:Nn \l__enumext_joined_item_viii_int {#1}
3185     \int_compare:nNnT { \l__enumext_joined_item_viii_int } > { \l__enumext_columns_viii_int }
3186     {
3187         \msg_warning:nnee { enumext } { item-joined }
3188         { \int_use:N \l__enumext_joined_item_viii_int }
3189         { \int_use:N \l__enumext_columns_viii_int }
3190         \int_set:Nn \l__enumext_joined_item_viii_int
3191         {
3192             \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + \c_one_int
3193         }
3194     }
3195     \int_compare:nNnT
3196     { \l__enumext_joined_item_viii_int }

```

```

3197     >
3198     { \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + \c_one_int }
3199     {
3200         \msg_warning:nnee { enumext } { item-joined-columns }
3201         { \int_use:N \l__enumext_joined_item_viii_int }
3202         {
3203             \int_eval:n
3204             { \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + \c_one_int }
3205         }
3206         \int_set:Nn \l__enumext_joined_item_viii_int
3207         {
3208             \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + \c_one_int
3209         }
3210     }
3211     \int_compare:nNnTF { \l__enumext_joined_item_viii_int } > { \c_one_int }
3212     {
3213         \int_set_eq:NN \l__enumext_joined_item_aux_viii_int \l__enumext_joined_item_viii_int
3214         \int_decr:N \l__enumext_joined_item_aux_viii_int
3215         \int_add:Nn \l__enumext_item_column_pos_viii_int { \l__enumext_joined_item_aux_viii_int }
3216         \int_gadd:Nn \g__enumext_item_count_all_viii_int { \l__enumext_joined_item_aux_viii_int }
3217         \dim_set:Nn \l__enumext_joined_width_viii_dim
3218         {
3219             \l__enumext_item_width_viii_dim * \l__enumext_joined_item_viii_int
3220             + ( \l__enumext_labelwidth_viii_dim + \l__enumext_labelsep_viii_dim
3221                 + \l__enumext_columns_sep_viii_dim
3222             ) * \l__enumext_joined_item_aux_viii_int
3223         }
3224         \dim_set_eq:NN \itemwidth \l__enumext_joined_width_viii_dim
3225     }
3226     {
3227         \dim_set_eq:NN \l__enumext_joined_width_viii_dim \l__enumext_item_width_viii_dim
3228         \dim_set_eq:NN \itemwidth \l__enumext_item_width_viii_dim
3229     }
3230 }

```

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

```

\__enumext_start_mini_viii:
\__enumext_stop_mini_viii:

```

The implementation of the `mini-env` key is identical to the one used in the `enumext*` environment.

```

3231 \cs_new_protected:Nn \__enumext_start_mini_viii:
3232 {
3233     \dim_compare:nNnT { \l__enumext_minipage_right_viii_dim } > { \c_zero_dim }
3234     {
3235         \dim_set:Nn \l__enumext_minipage_left_viii_dim
3236         {
3237             \linewidth
3238             - \l__enumext_minipage_right_viii_dim
3239             - \l__enumext_minipage_hsep_viii_dim
3240         }
3241         \bool_set_true:N \l__enumext_minipage_active_viii_bool
3242         \dim_gset_eq:NN
3243         \g__enumext_minipage_right_viii_dim
3244         \l__enumext_minipage_right_viii_dim
3245         \__enumext_mini_addvspace_viii:
3246         \nointerlineskip\noindent
3247         \begin{__enumext_mini_env*}{ \l__enumext_minipage_left_viii_dim }
3248     }
3249 }
3250 \cs_new_protected:Nn \__enumext_stop_mini_viii:
3251 {
3252     \bool_if:NT \l__enumext_minipage_active_viii_bool
3253     {
3254         \end{__enumext_mini_env*}
3255         \hfill
3256         \bool_gset_true:N \g__enumext_minipage_active_viii_bool
3257     }
3258 }
3259 \__enumext_after_env:n {keyans*}
3260 {
3261     \bool_if:NT \g__enumext_minipage_active_viii_bool
3262     {
3263         \begin{__enumext_mini_env*}{ \g__enumext_minipage_right_viii_dim }

```

```

3264         \par\addvspace { \g__enumext_minipage_right_skip }
3265         \bool_if:NF \g__enumext_minipage_center_viii_bool
3266         {
3267             \centering
3268         }
3269         \tl_use:N \g__enumext_miniright_code_viii_tl % the code
3270         \end{__enumext_mini_env*}
3271         \par\addvspace{ \g__enumext_minipage_after_skip }
3272     }
3273     \bool_gset_false:N \g__enumext_minipage_active_viii_bool
3274     \bool_gset_true:N \g__enumext_minipage_center_viii_bool
3275     \tl_gclear:N \g__enumext_miniright_code_viii_tl
3276     \dim_gzero:N \g__enumext_minipage_right_viii_dim
3277 }

```

(End of definition for `__enumext_start_mini_viii:` and `__enumext_stop_mini_viii:`.)

keyans* First we will generate the environment and we will give a temporary definition to `__enumext_stop_item_tmp_viii:` equal to `\noindent` and next to `\item` equal to `__enumext_start_item_tmp_viii:` which we will redefine later.

```

3278 \NewDocumentEnvironment{keyans*}{ o }
3279 {
3280     \__enumext_safe_exec_viii:
3281     \__enumext_parse_keys_viii:n {#1}
3282     \__enumext_before_list_viii:
3283     \__enumext_start_list:nn { }
3284     {
3285         \__enumext_list_arg_two_viii:
3286         \__enumext_before_keys_exec_viii:
3287     }
3288     \__enumext_starred_columns_set_viii:
3289     \item[] \scan_stop:
3290     \cs_set_eq:NN \__enumext_stop_item_tmp_viii: \noindent
3291     \cs_set_eq:NN \item \__enumext_start_item_tmp_viii:
3292 }
3293 {
3294     \__enumext_stop_item_tmp_viii:
3295     \__enumext_remove_extra_parsep_viii:
3296     \__enumext_stop_list:
3297     \__enumext_after_list_viii:
3298 }

```

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

`__enumext_safe_exec_viii:` First check the maximum nesting level for the **keyans*** environment.

```

3299 \cs_new_protected:Nn \__enumext_safe_exec_viii:
3300 {
3301     \int_incr:N \l__enumext_keyans_level_h_int
3302     \int_compare:nNnT { \l__enumext_keyans_level_h_int } > { 1 }
3303     {
3304         \msg_error:nn { enumext } { nested }
3305     }
3306     % Set false for interfering with enumext nested in keyans* (yes, its possible and crayze)
3307     \bool_set_false:N \l__enumext_store_active_bool
3308     \int_compare:nNnT { \l__enumext_level_int } > { 1 }
3309     {
3310         \msg_error:nn { enumext } { keyans-wrong-level }
3311     }
3312 }

```

(End of definition for `__enumext_safe_exec_viii:`.)

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

```

3313 \cs_new_protected:Npn \__enumext_parse_keys_viii:n #1
3314 {
3315     \tl_if_novalue:nF {#1}
3316     {
3317         \keys_set:nn { enumext / keyans* } {#1}
3318     }
3319 }

```

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

`__enumext_before_list_viii:` The function `__enumext_before_list_viii:` will add the vertical spacing on the environment if the `above` key is active next to the `{\code}` defined by the `before*` key if it is active, the call the function `__enumext_start_mini_viii:` handle by `mini-env`.

```
3320 \cs_new_protected:Nn \__enumext_before_list_viii:
3321 {
3322     \__enumext_vspace_above_viii:
3323     \__enumext_before_args_exec_viii:
3324     \__enumext_start_mini_viii:
3325 }
```

(End of definition for `__enumext_before_list_viii:.`)

`__enumext_after_list_viii:` The function `__enumext_after_list:` first call the function `__enumext_stop_mini_viii:`, then apply the `{\code}` handled by the `after` key together with the *vertical space* handled by the `below` key if they are present.

```
3326 \cs_new_protected:Nn \__enumext_after_list_viii:
3327 {
3328     \__enumext_stop_mini_viii:
3329     \__enumext_after_stop_list_viii:
3330     \__enumext_vspace_below_viii:
3331 }
```

(End of definition for `__enumext_after_list_viii:.`)

10.36.2 The command `\item` in `keyans*`

The idea here is to make the `\item` command behave in the same way as in the `keyans` environment with the difference of the optional argument (`\number`) which works in the same way as in the `enumext*` environment. In simple terms we want to store the `\label` next to the `[\content]` if it is present in the `\sequence` and `\prop list` defined by `save-ans` key for `\item*`, `\item*[\content]`, `\item(\number)*` and `\item(\number)*[\content]` commands.

`__enumext_start_item_tmp_viii:` First we will call the function `__enumext_stop_item_tmp_viii:` that we will redefine later, we will increment the value of `__enumext_item_column_pos_viii_int` that will count the item's by rows and the value of `\g__enumext_item_count_all_viii_int` that will count the total of item's in the environment. After that we will call the function `__enumext_item_peek_args_viii:` that will handle the arguments passed to `\item`.

```
3332 \cs_new_protected_nopar:Nn \__enumext_start_item_tmp_viii:
3333 {
3334     \__enumext_stop_item_tmp_viii:
3335     \int_incr:N \__enumext_item_column_pos_viii_int
3336     \int_gincr:N \g__enumext_item_count_all_viii_int
3337     \__enumext_item_peek_args_viii:
3338 }
```

(End of definition for `__enumext_start_item_tmp_viii:.`)

`__enumext_item_peek_args_viii:` The function `__enumext_item_peek_args_viii:` will handle the `\item(\number)`. Look for the argument “(”, if it is present we will call the function `__enumext_joined_item_viii:w` (`\number`), which is in charge of joining the item's in the same row, in case they are not present we will set the default value (1).

```
3339 \cs_new_protected:Nn \__enumext_item_peek_args_viii:
3340 {
3341     \peek_meaning:NTF (
3342         { \__enumext_joined_item_viii:w }
3343         { \__enumext_joined_item_viii:w (1) }
3344     }
```

(End of definition for `__enumext_item_peek_args_viii:.`)

`__enumext_joined_item_viii:w` The function `__enumext_joined_item_viii:w` will first call the function `__enumext_starred_joined_item_viii:n` in charge of setting the *width* of the box that will store the content passed to `\item`. Then we will look for the argument “*”, if it is present we will call the function `__enumext_starred_item_viii:w` otherwise we will call the function `__enumext_standard_item_viii:w`.

```
3345 \cs_new_protected:Npn \__enumext_joined_item_viii:w (#1)
3346 {
3347     \__enumext_starred_joined_item_viii:n {#1}
3348     \peek_meaning_remove:NTF *
3349     { \__enumext_starred_item_viii:w }
3350     { \__enumext_standard_item_viii:w }
3351 }
```

(End of definition for `__enumext_joined_item_viii:w`)

`__enumext_standard_item_viii:w`

The function `__enumext_standard_item_viii:w` will first look for the argument “[”, if present it will set the state of the variable `\l__enumext_wrap_label_opt_viii_bool` equal to the state of the variable `\l__enumext_wrap_label_opt_viii_bool` handled by the key `wrap-label*` and finally execute the *non-enumerated* version `\item[⟨custom⟩]` by means of the function `__enumext_start_item_viii:w`, otherwise we will set the value of the variable `\l__enumext_wrap_label_viii_bool` handled by the `wrap-label` key to true and set the switch `\if@noitemarg` to true to execute the enumerated version of `\item` by means of the function `__enumext_start_item_viii:w [\l__enumext_label_viii_tl]`.

```

3352 \cs_new_protected:Npn \__enumext_standard_item_viii:w
3353 {
3354   \bool_set_false:N \l__enumext_item_starred_viii_bool
3355   \peek_meaning:NTF [
3356     {
3357       \bool_set_eq:NN
3358         \l__enumext_wrap_label_viii_bool
3359         \l__enumext_wrap_label_opt_viii_bool
3360       \__enumext_start_item_viii:w
3361     }
3362     {
3363       \bool_set_true:N \l__enumext_wrap_label_viii_bool
3364       \legacy_if_set_true:n { @noitemarg }
3365       \__enumext_start_item_viii:w [ \l__enumext_label_viii_tl ]
3366     }
3367   }

```

(End of definition for `__enumext_standard_item_viii:w`)

`__enumext_starred_item_viii:w`
`__enumext_starred_item_viii_aux_i:w`
`__enumext_starred_item_viii_aux_ii:w`

The function `__enumext_starred_item_viii:w` together with the specified auxiliary functions `aux_i:w` and `aux_ii:w` execute `\item*` and `\item*[⟨content⟩]`.

```

3368 \cs_new_protected:Npn \__enumext_starred_item_viii:w
3369 {
3370   \bool_set_true:N \l__enumext_item_starred_viii_bool
3371   \bool_set_true:N \l__enumext_wrap_label_viii_bool
3372   \peek_meaning:NTF [
3373     { \__enumext_starred_item_viii_aux_i:w }
3374     { \__enumext_starred_item_viii_aux_ii:w }
3375   }
3376   \cs_new_protected:Npn \__enumext_starred_item_viii_aux_i:w [#1]
3377   {
3378     \tl_clear:N \l__enumext_store_keyans_label_tl
3379     \tl_if_novalue:nF { #1 }
3380     {
3381       \tl_set:Nx \l__enumext_store_keyans_label_tl { \c_space_tl #1 }
3382     }
3383     \__enumext_starred_item_viii_aux_ii:w
3384   }
3385   \cs_new_protected:Npn \__enumext_starred_item_viii_aux_ii:w
3386   {
3387     \legacy_if_set_true:n { @noitemarg }
3388     \__enumext_start_item_viii:w [ \l__enumext_label_viii_tl ]
3389     \tl_put_left:Nx \l__enumext_store_keyans_label_tl { \l__enumext_label_viii_tl }
3390     \__enumext_store_addto_prop:V \l__enumext_store_keyans_label_tl
3391     \__enumext_keyans_store_ref:
3392     \tl_put_left:Nx \l__enumext_store_keyans_label_tl { \item }
3393     \__enumext_keyans_addto_seq_link:
3394   }

```

(End of definition for `__enumext_starred_item_viii:w`, `__enumext_starred_item_viii_aux_i:w`, and `__enumext_starred_item_viii_aux_ii:w`)

Real definition of `\item`

The functions `__enumext_start_item_viii:w` and `__enumext_stop_item_viii:` executing the true definition of `\item` inside the `keyans*` environment.

`__enumext_start_item_viii:w`

The first thing we will do is set the value of `__enumext_stop_item_tmp_viii:` equal to the value of `__enumext_stop_item_viii:` which we will define later and add the `hyperref` compatible `enumXviii` 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.

```

3395 \cs_new_protected_nopar:Npn \__enumext_start_item_viii:w [#1]
3396 {
3397   \cs_set_eq:NN \__enumext_stop_item_tmp_viii: \__enumext_stop_item_viii:
3398   \legacy_if:nT { @noitemarg }
3399   {
3400     \legacy_if_set_false:n { @noitemarg }
3401     \legacy_if:nT { @nmbrlist }
3402     {
3403       \bool_if:NT \l__enumext_hyperref_bool
3404       {
3405         \legacy_if_set_true:n { @hyper@item }
3406       }
3407       \refstepcounter{enumXviii}
3408     }
3409   }

```

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

```

3410 \group_begin:
3411   \lrbox{ \l__enumext_item_text_viii_box }
3412   \bool_if:NF \l__enumext_footnotes_key_bool
3413   {
3414     \__enumext_renew_footnote:
3415   }
3416   \bool_if:NT \l__enumext_item_starred_viii_bool
3417   {
3418     %\tl_if_blank:VT \g__enumext_item_symbol_aux_viii_tl
3419     %{
3420       %\tl_gset_eq:NN
3421       %\g__enumext_item_symbol_aux_viii_tl \l__enumext_item_symbol_viii_tl
3422     %}
3423     \mode_leave_vertical:
3424     %%\skip_horizontal:n { -\l__enumext_item_symbol_sep_viii_dim }
3425     %\makebox[ 0pt ][ r ]{ \g__enumext_item_symbol_aux_viii_tl }
3426     %%\skip_horizontal:N \l__enumext_item_symbol_sep_viii_dim
3427     %\tl_gclear:N \g__enumext_item_symbol_aux_viii_tl
3428   }
3429   \group_begin:
3430     \tl_use:N \l__enumext_label_font_style_viii_tl
3431     \bool_if:NTF \l__enumext_wrap_label_viii_bool
3432     {
3433       \makebox[ \l__enumext_labelwidth_viii_dim ][ \l__enumext_align_label_viii_str ]
3434       { \__enumext_wrapper_label_viii:n {#1} }
3435     }
3436     {
3437       \makebox[ \l__enumext_labelwidth_viii_dim ][ \l__enumext_align_label_viii_str ]{ #1
3438     }
3439   \group_end:
3440   \skip_horizontal:N \l__enumext_labelsep_viii_dim
3441   \tl_use:N \l__enumext_after_list_args_viii_tl
3442   \__enumext_minipage:w [ t ]{ \l__enumext_joined_width_viii_dim }
3443   \skip_set_eq:NN \parindent \l__enumext_listparindent_viii_dim
3444   \skip_set_eq:NN \parskip \l__enumext_parsep_viii_skip
3445   \tl_use:N \l__enumext_fake_item_indent_viii_tl
3446 }

```

(End of definition for `__enumext_start_item_viii:w`)

`__enumext_stop_item_viii:` The function `__enumext_stop_item_viii:` shall terminate with the capture of `\item` and its *(contents)*. Close the environments `minipage`, `lrbox` and the group. Then we only have to set the width of the box and print it next to `\footnote`, and add the horizontal and vertical separation between the boxes.

```

3447 \cs_new_protected_nopar:Nn \__enumext_stop_item_viii:
3448 {
3449   \__enumext_endminipage:
3450   \endlrbox
3451   \group_end:
3452   \box_set_wd:Nn \l__enumext_item_text_viii_box
3453   {
3454     \l__enumext_joined_width_viii_dim
3455     + \l__enumext_labelwidth_viii_dim

```

```

3456         + \l__enumext_labelsep_viii_dim
3457     }
3458     \int_set:Nn \hbadness { 10000 }
3459     \box_use:N \l__enumext_item_text_viii_box
3460     \bool_if:NF \l__enumext_footnotes_key_bool
3461     {
3462         \__enumext_print_footnote:
3463     }
3464     \int_compare:nNnTF { \l__enumext_item_column_pos_viii_int } = { \l__enumext_columns_viii_int }
3465     {
3466         \par\noindent
3467         \int_zero:N \l__enumext_item_column_pos_viii_int
3468     }
3469     { \hspace{ \l__enumext_columns_sep_viii_dim } }
3470 }

```

(End of definition for __enumext_stop_item_viii:.)

__enumext_remove_extra_parsep_viii:

Finally we will remove the vertical space equal to `\parsep` when the total number of items is divisible by the number of items in the last row of the environment.

```

3471 \cs_new_protected:Nn \__enumext_remove_extra_parsep_viii:
3472 {
3473     \int_compare:nNnTF
3474     {
3475         \int_mod:nn { \g__enumext_item_count_all_viii_int } { \l__enumext_columns_viii_int }
3476     }
3477     =
3478     { \c_zero_int }
3479     {
3480         \par
3481         \vspace{ -\l__enumext_itemsep_viii_skip }
3482         \int_gzero:N \g__enumext_item_count_all_viii_int
3483     }
3484 }

```

(End of definition for __enumext_remove_extra_parsep_viii:.)

10.37 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.

```

3485 \NewDocumentCommand \getkeyans { m }
3486 {
3487     \exp_args:Ne \__enumext_getkeyans_aux:n
3488     { \tl_to_str:e { \text_expand:n {#1} } }
3489 }

```

(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.

```

3490 \cs_new_protected:Npn \__enumext_getkeyans_aux:n #1
3491 {
3492     \str_if_in:nnTF {#1} { : }
3493     {
3494         \use:e
3495         {
3496             \cs_set:Npn \exp_not:N \__enumext_tmp:w ##1 \c_colon_str ##2 \scan_stop:
3497             { {##1} {##2} }
3498         }
3499         \exp_after:wN \__enumext_getkeyans:nn \__enumext_tmp:w #1 \scan_stop:
3500     }
3501     { \msg_error:nnn { enumext } { missing-colon } {#1} }
3502 }

```

(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 $\langle prop\ list \rangle$.

```

3503 \cs_new_protected:Npn \__enumext_getkeyans:nn #1 #2

```

```

3504 {
3505   \prop_if_exist:cF { g__enumext_#1_prop }
3506   { \msg_error:nnn { enumext } { undefined-storage-anskey } {#1} }
3507   \group_begin:
3508   \prop_item:cn { g__enumext_#1_prop }{#2}
3509   \group_end:
3510 }

```

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

10.38 The command `\printkeyans`

The `\printkeyans` command prints “all stored content” in the *sequence* defined by the `save-ans` key. The first thing we will do is to define a set of *keys* 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`.

```

3511 \keys_define:nn { keyanskey / print }
3512 {
3513   level-1 .code:n = \tl_put_right:Nn \l__enumext_print_keyans_i_tl
3514   {
3515     \setenumext[level,1] {#1} \setenumext[print,1] {#1}
3516   },
3517   level-1 .initial:n = { label=\arabic*., nosep, columns=2, first=\small, font=\small },
3518   level-2 .code:n = \tl_put_right:Nn \l__enumext_print_keyans_ii_tl
3519   {
3520     \setenumext[level,2] {#1} \setenumext[print,2] {#1}
3521   },
3522   level-2 .initial:n = { nosep, label=(\alph*), first=\small, font=\small },
3523   level-3 .code:n = \tl_put_right:Nn \l__enumext_print_keyans_iii_tl
3524   {
3525     \setenumext[level,3] {#1} \setenumext[print,3] {#1}
3526   },
3527   level-3 .initial:n = { nosep, label=\roman*., first=\small, font=\small },
3528   level-4 .code:n = \tl_put_right:Nn \l__enumext_print_keyans_iv_tl
3529   {
3530     \setenumext[level,4] {#1} \setenumext[print,4] {#1}
3531   },
3532   level-4 .initial:n = { nosep, label=\Alph*., first=\small, font=\small },
3533   level-* .code:n = \tl_put_right:Nn \l__enumext_print_keyans_vii_tl % starred
3534   {
3535     \setenumext[enumext*] {#1} %%\setenumext[print,*] {#1}
3536   },
3537   level-* .initial:n = { label=\arabic*., nosep, columns=2, first=\small, font=\small },
3538 }

```

`\printkeyans` Create a user command to print “all stored content” in *sequence* for `\anskey`, `\item*` and `\anspic*`.

```

3539 \NewDocumentCommand \printkeyans { s O{ } m }
3540 {
3541   \group_begin:
3542   \tl_use:N \l__enumext_print_keyans_i_tl
3543   \tl_use:N \l__enumext_print_keyans_ii_tl
3544   \tl_use:N \l__enumext_print_keyans_iii_tl
3545   \tl_use:N \l__enumext_print_keyans_iv_tl
3546   \tl_use:N \l__enumext_print_keyans_vii_tl
3547   \__enumext_printkeyans:nnn { #1 } { #2 } { #3 }
3548   \group_end:
3549 }

```

(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

```

3550 \cs_new_protected:Npn \__enumext_printkeyans:nnn #1 #2 #3
3551 {
3552   \seq_if_exist:cTF { g__enumext_#3_seq }
3553   {

```



```

3554 \seq_if_empty:cF { g__enumext_#3_seq }
3555 {
3556   %%\seq_show:c { g__enumext_#3_seq }
3557   \bool_if:nTF {#1}
3558   {
3559     \begin{enumext*}[#2]
3560     \seq_map_inline:cn { g__enumext_#3_seq } { ##1 }
3561     \end{enumext*}
3562   }
3563   {
3564     \begin{enumext}[#2]
3565     \seq_map_inline:cn { g__enumext_#3_seq } { ##1 }
3566     \end{enumext}
3567   }
3568 }
3569 }
3570 {
3571   \msg_error:nnn { enumext } { undefined-storage-anskey } {#3}
3572 }
3573 }

```

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

10.39 The command `\setenumext`

First we define a “meta families” of *(keys)* to access from `\setenumext`.

```

3574 \keys_define:nn { enumext / meta-families }
3575 {
3576   level-1 .code:n = { \keys_set:nn { enumext / level-1 } {#1} } ,
3577   level-2 .code:n = { \keys_set:nn { enumext / level-2 } {#1} } ,
3578   level-3 .code:n = { \keys_set:nn { enumext / level-3 } {#1} } ,
3579   level-4 .code:n = { \keys_set:nn { enumext / level-4 } {#1} } ,
3580   keyans .code:n = { \keys_set:nn { enumext / keyans } {#1} } ,
3581   enumext* .code:n = { \keys_set:nn { enumext / enumext* } {#1} } ,
3582   keyans* .code:n = { \keys_set:nn { enumext / keyans* } {#1} } ,
3583   print-1 .code:n = { \keys_set:nn { keyanskey / print } { level-1 = {#1} } } ,
3584   print-2 .code:n = { \keys_set:nn { keyanskey / print } { level-2 = {#1} } } ,
3585   print-3 .code:n = { \keys_set:nn { keyanskey / print } { level-3 = {#1} } } ,
3586   print-4 .code:n = { \keys_set:nn { keyanskey / print } { level-4 = {#1} } } ,
3587   print-* .code:n = { \keys_set:nn { keyanskey / print } { level-* = {#1} } } ,
3588   unknown .code:n = { \msg_error:nn { enumext } { unknown-key-family } } ,
3589 }

```

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

```

3590 \seq_const_from_clist:Nn \c__enumext_all_families_seq
3591 {
3592   level-1 , level-2 , level-3 , level-4 , keyans , enumext* ,
3593   keyans* , print-1 , print-2 , print-3 , print-4 , print-* ,
3594 }

```

`\setenumext` Now we define the user command `\setenumext`.

```

3595 \NewDocumentCommand \setenumext { o +m }
3596 {
3597   \tl_if_novalue:nTF {#1}
3598   {
3599     \seq_map_inline:Nn \c__enumext_all_families_seq
3600     }
3601   {
3602     \seq_clear:N \l__enumext_setkey_tmpa_seq
3603     \seq_set_from_clist:Nn \l__enumext_setkey_tmpb_seq {#1}
3604     \int_set:Nn \l__enumext_setkey_tmpa_int
3605     {
3606       \seq_count:N \l__enumext_setkey_tmpb_seq
3607     }
3608     \int_compare:nNnTF { \l__enumext_setkey_tmpa_int } > { 1 }
3609     {
3610       \seq_pop_left:NN \l__enumext_setkey_tmpb_seq \l__enumext_setkey_tmpa_tl
3611       \seq_map_function:NN \l__enumext_setkey_tmpb_seq \__enumext_set_parse:n
3612       \seq_set_map_e:NNn \l__enumext_setkey_tmpa_seq \l__enumext_setkey_tmpa_seq
3613       {
3614         \tl_use:N \l__enumext_setkey_tmpa_tl - ##1
3615       }

```

```

3616     }
3617     {
3618         \seq_put_right:Ne \l__enumext_setkey_tmpa_seq { \tl_trim_spaces:n {#1} }
3619     }
3620     \seq_if_empty:NTF \l__enumext_setkey_tmpa_seq
3621     { \seq_map_inline:Nn \c__enumext_all_families_seq }
3622     { \seq_map_inline:Nn \l__enumext_setkey_tmpa_seq }
3623 }
3624 {
3625     \keys_set:nn { enumext / meta-families } { ##1 = {#2} }
3626 }
3627 }

```

(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.

```

3628 \cs_new_protected:Npn \__enumext_set_parse:n #1
3629 {
3630     \tl_set:Ne \l__enumext_setkey_tmpb_tl { \tl_trim_spaces:n {#1} }
3631     \int_step_inline:nnn { 0 } { 4 } % <- max level
3632     { \tl_remove_all:Nn \l__enumext_setkey_tmpb_tl {##1} }
3633     \tl_if_empty:NTF \l__enumext_setkey_tmpb_tl
3634     {
3635         \seq_put_right:Ne \l__enumext_setkey_tmpa_seq
3636         { \tl_trim_spaces:n {#1} }
3637     }
3638     { \__enumext_set_error:nn {#1} { } }
3639 }
3640 \cs_new_protected:Npn \__enumext_set_error:nn #1 #2
3641 { \msg_error:nnn { enumext } { invalid-key } {#1} {#2} }

```

(End of definition for `__enumext_set_parse:n` and `__enumext_set_error:nn`.)

10.40 Messages

Message used by package-load for `multicol` and `hyperref` packages.

```

3642 \msg_new:nnn { enumext } { package-load }
3643 {
3644     The ~ '#1' ~ package ~ is ~ already ~ loaded.
3645 }
3646 \msg_new:nnn { enumext } { package-not-load }
3647 {
3648     The ~ '#1' ~ package ~ will ~ be ~ loaded ~ as ~ a ~ dependency.
3649 }
3650 \msg_new:nnn { enumext } { package-load-foot }
3651 {
3652     The ~ '#1' ~ package ~ is ~ loaded ~ with ~ the ~ option ~ '#2'.
3653 }

```

Message used in the creation of counters by `enumext` package.

```

3654 \msg_new:nnn { enumext } { counters }
3655 {
3656     The ~ counter ~ '#1' ~ is ~ already ~ defined ~ by ~ some ~ \
3657     package ~ or ~ macro, ~ it ~ cannot ~ be ~ continued.
3658 }

```

Message used by `[\langle key = val \rangle]` system and `\setenumext` command.

```

3659 \msg_new:nnn { enumext } { invalid-key }
3660 {
3661     The ~ key ~ '#1' ~ is ~ not ~ know ~ the ~ level ~ #2.
3662 }
3663 \msg_new:nnn { enumext } { unknown-key-family }
3664 {
3665     Unknown~key~family~`\l_keys_key_str'~for~enumext.
3666 }

```

Messages used in length calculation.

```

3667 \msg_new:nnn { enumext } { width-negative }
3668 {
3669     Ignoring ~ negative ~ value ~ '#1=#2' ~ \msg_line_context:.\
3670     The ~ key ~ '#1' ~ accepts ~ values ~ >= ~ opt.

```

```

3671 }
3672 \msg_new:nnn { enumext } { width-zero }
3673 {
3674   Invalid ~ '#1=#2' ~ \msg_line_context:.\
3675   The ~ key ~ '#1'~ accepts ~ values ~ > ~ Opt.
3676 }

```

Messages used by `show-length` key in `enumext`.

```

3677 \msg_new:nnn { enumext } { list-lengths }
3678 {
3679   **** ~ Lengths ~ used ~ by ~ 'enumext' ~ level ~ '#2' ~ \msg_line_context:~\c_space_tl ****\\
3680   \__enumext_show_length:nnn { dim } { labelsep } {#1}
3681   \__enumext_show_length:nnn { dim } { labelwidth } {#1}
3682   \__enumext_show_length:nnn { dim } { itemindent } {#1}
3683   \__enumext_show_length:nnn { dim } { leftmargin } {#1}
3684   \__enumext_show_length:nnn { dim } { rightmargin } {#1}
3685   \__enumext_show_length:nnn { dim } { listparindent } {#1}
3686   \__enumext_show_length:nnn { skip } { topsep } {#1}
3687   \__enumext_show_length:nnn { skip } { parsep } {#1}
3688   \__enumext_show_length:nnn { skip } { partopsep } {#1}
3689   \__enumext_show_length:nnn { skip } { itemsep } {#1}
3690   ****
3691 }

```

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

```

3692 \msg_new:nnn { enumext } { list-lengths-not-nested }
3693 {
3694   **** ~ Lengths ~ used ~ by ~ '#2' ~ environment ~ \msg_line_context:~\c_space_tl ****\\
3695   \__enumext_show_length:nnn { dim } { labelsep } {#1}
3696   \__enumext_show_length:nnn { dim } { labelwidth } {#1}
3697   \__enumext_show_length:nnn { dim } { itemindent } {#1}
3698   \__enumext_show_length:nnn { dim } { leftmargin } {#1}
3699   \__enumext_show_length:nnn { dim } { rightmargin } {#1}
3700   \__enumext_show_length:nnn { dim } { listparindent } {#1}
3701   \__enumext_show_length:nnn { skip } { topsep } {#1}
3702   \__enumext_show_length:nnn { skip } { parsep } {#1}
3703   \__enumext_show_length:nnn { skip } { partopsep } {#1}
3704   \__enumext_show_length:nnn { skip } { itemsep } {#1}
3705   ****
3706 }

```

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

```

3707 \msg_new:nnn { enumext } { items-same-answer }
3708 {
3709   *****~Checking~answers~on~'#1'~OK~*****\\
3710   **~ All ~ items ~ stored ~ in ~ sequence ~ '#1' ~ have ~ an ~ answer. \\
3711   *****
3712   \prg_replicate:nn { 7 + \str_count:n {#1} } { * }
3713 }
3714 \msg_new:nnn { enumext } { item-different-answer }
3715 {
3716   Number ~ of ~ items ~ different ~ of ~ number ~ of ~
3717   answer ~ in ~ sequence ~ '#1'~ closed ~ \msg_line_context:.
3718 }

```

Messages used by the internal system to check for “starred” `\item*` commands.

```

3719 \msg_new:nnn { enumext } { missing-starred }
3720 {
3721   Missing ~ '\c_backslash_str #1*' ~ in ~ '#2' ~ \msg_line_context:.
3722 }

```

Message for the nesting depth of the environment `enumext`.

```

3723 \msg_new:nnn { enumext } { list-too-deep }
3724 {
3725   Too ~ deep ~ nesting ~ for ~ 'enumext' ~ \msg_line_context:~ \\
3726   The ~ maximum ~ level ~ of ~ nesting ~ is ~ 4.
3727 }

```

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

```

3728 \msg_new:nnn { enumext } { anskey-wrong-place }
3729 {
3730   Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:~ \\
3731   '\c_backslash_str #1' ~ works ~ in ~ the ~ environment ~ '#2'.
3732 }

```

```

3733 \msg_new:nnn { enumext } { anspic-wrong-place }
3734 {
3735   Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:~ \\
3736   '\c_backslash_str #1' ~ works ~ in ~ the ~ environment ~ '#2'.
3737 }
3738 \msg_new:nnn { enumext } { command-wrong-place }
3739 {
3740   Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:~ \\
3741   '\c_backslash_str #1' ~ works ~ outside ~ the ~ environment ~ '#2'.
3742 }

```

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

```

3743 \msg_new:nnn { enumext } { keyans-nested }
3744 {
3745   The ~ environment ~ 'keyans' ~ can't ~ be ~ nested ~ \msg_line_context:.
3746 }
3747 \msg_new:nnn { enumext } { keyans-wrong-level }
3748 {
3749   Wrong ~ level ~ position ~ for ~ 'keyans' ~ \msg_line_context:~ \\
3750   The ~ environment ~ 'keyans' ~ can ~ only ~ be ~ in ~ the ~ first ~ level.
3751 }
3752 \msg_new:nnn { enumext } { wrong-place }
3753 {
3754   Wrong ~ place ~ for ~ '#1' ~ environment ~ \msg_line_context:~ \\
3755   '#1' ~ is ~ only ~ found ~ with ~ '#2' ~ in ~ 'enumext'.
3756 }
3757 \msg_new:nnn { enumext } { keyanspic-nested }
3758 {
3759   The ~ environment ~ 'keyanspic' ~ can't ~ be ~ nested ~ \msg_line_context:~.
3760 }
3761 \msg_new:nnn { enumext } { keyanspic-wrong-level }
3762 {
3763   Wrong ~ level ~ position ~ for ~ 'keyanspic' ~ \msg_line_context:~ \\
3764   The ~ environment ~ 'keyans' ~ can ~ only ~ be ~ in ~ the ~ first ~ level.
3765 }

```

Messages used by `\getkeyans` command.

```

3766 \msg_new:nnn { enumext } { undefined-storage-anskey }
3767 {
3768   Storage ~ named ~ '#1' ~ is ~ not ~ defined ~ \msg_line_context:.
3769 }

```

Messages used by `\miniright` command.

```

3770 \msg_new:nnn { enumext } { missing-miniright }
3771 {
3772   Missing ~ '\c_backslash_str miniright' ~ in ~ \msg_line_context:.\
3773   The ~ key ~ 'mini-env' ~ need ~ '\c_backslash_str miniright'.
3774 }
3775 \msg_new:nnn { enumext } { wrong-miniright-place }
3776 {
3777   Wrong ~ place ~ for ~ '\c_backslash_str miniright' ~ \msg_line_context:~ \\
3778   Works ~ in ~ 'enumext' ~ and ~ 'keyans' ~ with ~ key ~ 'mini-env'.
3779 }
3780 \msg_new:nnn { enumext } { wrong-miniright-use }
3781 {
3782   Wrong ~ use ~ for ~ '\c_backslash_str miniright' ~ \msg_line_context:~ \\
3783   '\c_backslash_str miniright' ~ need ~ a ~ key ~ 'mini-env'.
3784 }

```

Messages used by `enumext*` and `keyans*` environments.

```

3785 \msg_new:nnn { enumext } { nested }
3786 {
3787   The ~ starred ~ environment ~ can't ~ be ~ nested ~ \msg_line_context:.
3788 }
3789 \msg_new:nnn { enumext } { item-joined }
3790 {
3791   Items ~ joined ~ (#1) ~ > ~ #2 ~ columns ~ \msg_line_context:.
3792 }
3793 \msg_new:nnn { enumext } { item-joined-columns }
3794 {
3795   Not ~ space ~ to ~ join ~ items ~ (#1) ~ > ~ #2 ~ \msg_line_context:.
3796 }

```

10.41 Finish package

Finish package implementation.

```
3797 \file_input_stop:
3798 \</package>
```

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>	400
<code>\+</code>	201
<code>\-</code>	201
<code>\\</code>	209, 2645, 3656, 3669, 3674, 3679, 3694, 3709, 3710, 3725, 3730, 3735, 3740, 3749, 3754, 3763, 3772, 3777, 3782
A	
<code>above</code>	<u>1211</u>
<code>above*</code>	<u>1211</u>
<code>\addvspace</code>	858, 886, 1009, 1088, 1151, 1157, 1185, 1202, 2462, 2484, 2601, 2616, 2840, 2847, 3264, 3271
<code>after</code>	<u>696</u>
<code>align</code>	<u>354</u>
<code>\Alph</code>	29, 33, 34
<code>\Alph</code>	306, 483, 501, 514, 3532
<code>\alph</code>	29, 33, 34
<code>\alph</code>	307, 481, 3522
<code>\anskey</code>	10, 57, <u>1639</u>
<code>\anspic</code>	12, 78, <u>2623</u>
<code>\arabic</code>	29, 31
<code>\arabic</code>	305, 480, 500, 3517, 3537
B	
<code>\b</code>	2351, 2364, 2909, 2922
<code>\baselineskip</code>	41
<code>\baselineskip</code>	1599, 1607
<code>before</code>	<u>696</u>
<code>before*</code>	<u>696</u>
<code>below</code>	<u>1211</u>
<code>below*</code>	<u>1211</u>
bool commands:	
<code>\bool_gset_false:N</code>	2507, 2849, 2957, 3160, 3273
<code>\bool_gset_true:N</code>	800, 2335, 2469, 2832, 2850, 2891, 2954, 3256, 3274
<code>\bool_if:NTF</code>	246, 258, 275, 1233, 1247, 1260, 1271, 1282, 1293, 1304, 1315, 1346, 1353, 1364, 1425, 1427, 1561, 1585, 1592, 1620, 1651, 1664, 1666, 1677, 1697, 1822, 1833, 1837, 1871, 1886, 1955, 1970, 1981, 2058, 2089, 2163, 2179, 2247, 2257, 2293, 2298, 2342, 2349, 2362, 2396, 2446, 2460, 2475, 2500, 2530, 2586, 2599, 2607, 2625, 2828, 2837, 2841, 2899, 2907, 2920, 2962, 2972, 3055, 3061, 3064, 3078, 3082, 3097, 3126, 3153, 3252, 3261, 3265, 3403, 3412, 3416, 3431, 3460
<code>\bool_if:nTF</code>	1186, 1203, 1705, 2101, 2135, 2199, 2646, 3557
<code>\bool_if_p:N</code>	2236, 2283
<code>\bool_lazy_all:nTF</code>	1405, 1754, 1763, 1776, 1792, 2329, 2885
<code>\bool_lazy_and:nnTF</code>	1687, 1730, 1940, 2234, 2282, 2378, 2465, 2952
<code>\bool_lazy_or:nnTF</code>	2651
<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, 118, 129, 130, 155, 166, 168
<code>\bool_not_p:n</code>	1689, 1781, 1796, 2331, 2380, 2467, 2887
<code>\bool_set_eq:NN</code>	2067, 2116, 3005, 3357
<code>\bool_set_false:N</code>	255, 1380, 1381, 1486, 1489, 1509, 1512, 2489, 2537, 2619, 2683, 2700, 2958, 3002, 3307, 3354
<code>\bool_set_true:N</code>	237, 241, 347, 624, 1217, 1222, 1341, 1360, 1371, 1485, 1488, 1508, 1511, 1521, 1528, 2063, 2094, 2112, 2124, 2328, 2384, 2389, 2415, 2535, 2561, 2817, 2884, 3011, 3018, 3019, 3241, 3363, 3370, 3371
box commands:	
<code>\box_dp:N</code>	905, 909, 913, 924, 928, 939, 948, 954, 964, 977, 983, 989, 1020, 1021, 1022, 1025, 1035, 1039, 1048, 1055, 1060, 1068, 1097, 1098, 1101, 1108, 1121, 1129, 1135, 1143, 2712
<code>\box_new:N</code>	43, 161
<code>\box_set_wd:Nn</code>	3118, 3452
<code>\box_use:N</code>	3125, 3459
<code>\box_wd:N</code>	313
C	
<code>\c</code>	400, 401, 524, 526, 538, 540
<code>\cB</code>	401
<code>\cE</code>	401
<code>\centering</code>	1188, 1205, 2733, 2843, 3267
<code>check-ans</code>	<u>1373</u>
Document class:	
<code>article</code>	35
clist commands:	
<code>\clist_const:Nn</code>	173
<code>\clist_map_function:nN</code>	2720
<code>\clist_map_inline:Nn</code>	353, 566, 629, 695, 710, 791, 1227
<code>\clist_map_inline:nn</code>	34, 51, 57, 69, 81, 104, 128, 138, 152, 172, 217, 378, 395, 634, 806, 1386, 1498, 1516, 1537, 1751, 1880, 2016, 2228, 2231, 2264, 2274, 2277, 2303
<code>\columnbreak</code>	58
<code>\columnbreak</code>	1691
<code>columns</code>	<u>775</u>
<code>columns*</code>	<u>1517</u>
<code>columns-sep</code>	<u>775</u>
<code>columns-sep*</code>	<u>1517</u>
<code>\columnsep</code>	73, 76
<code>\columnsep</code>	2440, 2583
<code>\columnseprule</code>	73, 77
<code>\columnseprule</code>	2444, 2585
Commands provide by enumext:	
<code>\anskey</code>	23, 24, 50–52, 55, 57, 59–61, 63, 72, 85, 95, 96, 99
<code>\anspic*</code>	23, 61–63, 78–80, 95, 96
<code>\anspic</code>	55, 78–80, 99
<code>\getkeyans</code>	55, 95, 100
<code>\item*</code>	23, 55, 61–63, 65, 66, 86, 93, 95, 96
<code>\itemwidth</code>	81, 89
<code>\item</code>	65, 66, 81, 85–87, 89, 92, 93
<code>\miniright</code>	23, 39, 46, 47, 73, 74, 76, 77, 100
<code>\printkeyans</code>	24, 55, 96
<code>\setenumext</code>	23, 97, 98
Counters defined by enumext:	
<code>enumXiii</code>	22, 29
<code>enumXii</code>	22, 29
<code>enumXiv</code>	22, 29
<code>enumXi</code>	22, 29
<code>enumXviii</code>	22, 29, 93
<code>enumXvii</code>	22, 29, 87
<code>enumXvi</code>	22, 29

enumXv	22, 29	\c_zero_dim	640, 654, 666, 678, 1178, 1196, 1717, 2189, 2194, 2200, 2207, 2407, 2430, 2555, 2573, 2741, 2809, 3165, 3233
cs commands:		E	
\cs_generate_variant:Nn	315, 331, 530, 546, 1551, 1558, 1638, 2218, 2722	\end ..	1181, 1199, 1587, 1622, 2459, 2483, 2598, 2615, 2830, 2846, 3254, 3270, 3561, 3566
\cs_if_exist:NTF	285	\endlist	27
\cs_new:Nn	186	\endlist	222
\cs_new:Npn	190, 195, 205	\endlrbox	3116, 3450
\cs_new_eq:NN	221, 222, 223, 227, 228, 260, 261, 264, 265	\endminipage	27
\cs_new_protected:Nn	232, 396, 416, 448, 711, 715, 719, 723, 727, 731, 735, 739, 743, 747, 751, 755, 759, 763, 767, 771, 807, 819, 843, 860, 871, 895, 970, 994, 1011, 1073, 1090, 1112, 1147, 1153, 1228, 1242, 1256, 1267, 1278, 1289, 1300, 1311, 1351, 1362, 1423, 1435, 1452, 1559, 1583, 1590, 1618, 1625, 1742, 1869, 1884, 1912, 1938, 2021, 2025, 2044, 2097, 2131, 2147, 2157, 2173, 2323, 2376, 2394, 2401, 2424, 2454, 2473, 2528, 2551, 2569, 2594, 2605, 2642, 2685, 2698, 2718, 2723, 2739, 2807, 2826, 2877, 2934, 2940, 2960, 2970, 2987, 3137, 3163, 3231, 3250, 3299, 3320, 3326, 3339, 3471	\endminipage	228
\cs_new_protected:Npn	178, 182, 268, 283, 300, 310, 316, 404, 423, 517, 531, 1175, 1194, 1338, 1391, 1542, 1552, 1674, 1819, 1831, 1853, 1922, 1960, 1968, 2054, 2073, 2108, 2120, 2187, 2221, 2267, 2338, 2347, 2547, 2693, 2758, 2894, 2905, 2993, 3000, 3016, 3024, 3029, 3041, 3182, 3313, 3345, 3352, 3368, 3376, 3385, 3490, 3503, 3550, 3628, 3640	enumext	4, 2304
\cs_new_protected_nopar:Nn	2980, 3113, 3332, 3447	enumext internal commands:	
\cs_new_protected_nopar:Npn	3047, 3395	__enumext_add_pre_parsep: ...	40, 817, 819, 819
\cs_set:Nn	193, 1824	__enumext_after_args_exec: .	37, 711, 723, 2316
\cs_set:Npn	1752, 1790, 3496	__enumext_after_args_exec_v: 38, 727, 739, 2521	
\cs_set_eq:NN	192, 197, 2867, 2868, 3049, 3290, 3291, 3397	__enumext_after_args_exec_vii: ...	743, 767
\cs_set_protected:Nn	211, 635, 651, 663, 675	__enumext_after_args_exec_viii:	771
\cs_set_protected:Npn	30, 44, 52, 64, 70, 94, 123, 134, 146, 153, 213, 332, 354, 383, 464, 484, 547, 567, 611, 630, 687, 696, 775, 792, 1211, 1373, 1470, 1499, 1517, 1744, 1873, 2005, 2219, 2265	__enumext_after_env:n	74
\cs_to_str:N	302, 325	__enumext_after_env:nn ..	75, 89, 182, 182, 2498, 2835, 3151, 3259
D		__enumext_after_hyperref: ...	27, 230, 232, 232
\d	201	__enumext_after_list: 74, 84, 92, 2321, 2473, 2473	
\DeclareDocumentEnvironment	888	\l__enumext_after_list_args_v_tl	741
dim commands:		\l__enumext_after_list_args_vii_tl	769, 3107
\dim_abs:n	2192, 2197	\l__enumext_after_list_args_viii_tl	773, 3441
\dim_add:Nn	2703	__enumext_after_list_v: ..	77, 2526, 2605, 2605
\dim_compare:nNnTF .	637, 653, 665, 677, 1177, 1196, 2189, 2194, 2200, 2206, 2208, 2210, 2406, 2429, 2555, 2573, 2695, 2741, 2809, 3165, 3233	__enumext_after_list_vii: ...	2875, 2940, 2940
\dim_compare:nTF	1715	__enumext_after_list_viii: ..	3297, 3326, 3326
\dim_gset_eq:NN	2818, 3242	__enumext_after_star_env:nn	82
\dim_gzero:N	2852, 3276	__enumext_after_stop_list: ...	37, 38, 711, 719, 2487
\dim_new:N .	40, 47, 48, 49, 66, 101, 111, 162, 163, 169	__enumext_after_stop_list_v: 38, 727, 735, 2620	
\dim_set:Nn ..	313, 625, 1529, 2087, 2192, 2197, 2199, 2202, 2203, 2207, 2209, 2212, 2213, 2215, 2409, 2432, 2557, 2575, 2725, 2743, 2750, 2793, 2811, 3043, 3167, 3174, 3217, 3235	\l__enumext_after_stop_list_v_tl	737
\dim_set_eq:NN	471, 491, 507, 511, 2082, 2230, 2276, 2366, 2440, 2583, 2800, 2803, 2804, 2924, 3034, 3224, 3227, 3228	__enumext_after_stop_list_vii: 743, 759, 2943	
\dim_use:N	638, 646, 1178, 1184, 1628, 1631, 1636, 2152, 2154, 2407, 2412, 2413, 2420, 2430, 2434, 2435, 2437	\l__enumext_after_stop_list_vii_tl ...	761
\dim_zero:N	2444, 2585, 2704, 2705, 2706	__enumext_after_stop_list_viii: .	763, 3329
\dim_zero_new:N	2756, 3180	\l__enumext_after_stop_list_viii_tl ...	765
		\l__enumext_align_label_vii_str ..	3099, 3103
		\l__enumext_align_label_viii_str .	3433, 3437
		\l__enumext_align_label_X_str	153
		\c__enumext_all_envs_clist ..	173, 353, 566, 629, 695, 710, 791, 1227
		\c__enumext_all_families_seq ..	97, 3590, 3599, 3621
		__enumext_anskey_wrapper:n	1474, 1829
		__enumext_at_begin_document:n ..	27, 178, 178, 219, 225
		__enumext_before_args_exec: 37, 711, 711, 2404	
		__enumext_before_args_exec_v: 37, 38, 727, 727, 2554	
		__enumext_before_args_exec_vii: ..	743, 743, 2937
		__enumext_before_args_exec_viii: 747, 3323	
		__enumext_before_keys_exec: 37, 711, 715, 2314	
		__enumext_before_keys_exec_v: ..	38, 727, 731, 2519
		__enumext_before_keys_exec_vii	743
		__enumext_before_keys_exec_vii: 38, 751, 2863	
		__enumext_before_keys_exec_viii: ..	38, 755, 3286
		__enumext_before_list: ...	72, 2308, 2401, 2401
		__enumext_before_list_v: .	76, 2514, 2551, 2551


```

\__enumext_before_list_vii: 84, 2858, 2934, 2934
\__enumext_before_list_viii: .. 92, 3282, 3320,
    3320
\l__enumext_before_no_starred_key_v_tl 733
\l__enumext_before_no_starred_key_vii_-
    tl ..... 753
\l__enumext_before_no_starred_key_viii_-
    tl ..... 757
\l__enumext_before_starred_key_v_tl ... 729
\l__enumext_before_starred_key_vii_tl . 745
\l__enumext_before_starred_key_viii_tl 749
\__enumext_calc_hspace:NNNNNN 68, 2187, 2187,
    2218, 2223, 2269
\__enumext_check_ans_active: .. 53, 1435, 1435,
    2504
\__enumext_check_ans_active_vii: . 1435, 1452,
    3157
\l__enumext_check_ans_bool ... 50, 65, 115, 1346,
    1377, 1381, 1425, 1666, 1955, 2058, 2089, 2466, 2952,
    3061
\__enumext_check_ans_count: . 52, 73, 1423, 1423,
    2405
\__enumext_check_ans_int:n .. 50, 52, 1348, 1391,
    1391
\g__enumext_check_ans_item_tl .. 62, 115, 1954,
    1962, 1966
\g__enumext_check_ans_show_bool 74, 115, 2469,
    2500, 2507
\g__enumext_check_ans_show_h_bool 115, 2954,
    3153, 3160
\l__enumext_columns_sep_v_dim 2573, 2575, 2583
\l__enumext_columns_sep_vii_dim .. 2741, 2743,
    2752, 2797, 2926, 3135
\l__enumext_columns_sep_viii_dim . 3165, 3167,
    3176, 3221, 3469
\l__enumext_columns_v_int 1016, 2571, 2579, 2591,
    2596
\l__enumext_columns_vii_int .. 2746, 2749, 2753,
    2761, 2765, 2768, 2774, 2780, 2784, 2913, 3130, 3141
\l__enumext_columns_viii_int . 3170, 3173, 3177,
    3185, 3189, 3192, 3198, 3204, 3208, 3464, 3475
\l__enumext_compare_items_ans_int 115, 1437,
    1443, 1454, 1461
\g__enumext_count_item_all_int 115, 1411, 1414,
    1439, 1456, 2060, 2091, 3070
\g__enumext_count_item_i_int ..... 1416, 1456
\g__enumext_count_item_ii_int 1417, 1439, 1457
\g__enumext_count_item_iii_int 1418, 1440, 1457
\g__enumext_count_item_iv_int 1419, 1440, 1458
\g__enumext_count_item_vii_int ..... 1420
\g__enumext_count_item_with_ans_int 52, 57, 62,
    115, 1421, 1443, 1461, 1668, 1957
\g__enumext_count_item_X_int ..... 115
\g__enumext_count_level_vii_int ..... 3071
\g__enumext_count_level_X_int ..... 115
\l__enumext_counter_i_tl ..... 30, 292
\l__enumext_counter_ii_tl ..... 30, 293
\l__enumext_counter_iii_tl ..... 30, 294
\l__enumext_counter_iv_tl ..... 30, 295
\l__enumext_counter_style_for_ref_vii_-
    tl ..... 431, 441, 452, 454
\l__enumext_counter_style_for_ref_viii_-
    tl ..... 458, 460
\l__enumext_counter_style_for_ref_X_tl 142
\c__enumext_counter_style_tl .... 31, 142, 398
\g__enumext_counter_styles_tl . 22, 29, 40, 303,
    321
\l__enumext_counter_v_tl ..... 30, 296
\l__enumext_counter_vi_tl ..... 30, 297
\l__enumext_counter_vii_tl ..... 30, 298, 428
\l__enumext_counter_viii_tl ..... 30, 299, 438
\l__enumext_current_widest_dim 22, 40, 327, 472,
    492, 508, 512
\__enumext_default_item:n ... 2054, 2054, 2105
\__enumext_define_counters:Nn 22, 283, 283, 292,
    293, 294, 295, 296, 297, 298, 299
\__enumext_endminipage: . 27, 225, 228, 894, 2735,
    3115, 3449
\__enumext_fake_item: ..... 635, 635, 2256
\l__enumext_fake_item_indent_v_dim 654, 659
\l__enumext_fake_item_indent_v_tl 656, 2113,
    2117, 2125
\l__enumext_fake_item_indent_vii_dim 666, 671
\l__enumext_fake_item_indent_vii_tl 668, 3111
\l__enumext_fake_item_indent_viii_dim . 678,
    683
\l__enumext_fake_item_indent_viii_tl .. 680,
    3445
\l__enumext_fake_item_indent_X_tl ..... 70
\__enumext_fake_item_vii: .... 635, 663, 2292
\__enumext_fake_item_viii: .... 635, 675, 2297
\g__enumext_footnote_arg_seq . 139, 2027, 2040,
    2050
\g__enumext_footnote_int . 139, 2034, 2037, 2039,
    2041
\g__enumext_footnote_int_seq . 139, 2028, 2041,
    2046, 2049
\__enumext_footnotes_key_bool ..... 27
\l__enumext_footnotes_key_bool 24, 27, 87, 129,
    241, 246, 255, 3078, 3126, 3412, 3460
\__enumext_footnotetext:nn ... 2021, 2021, 2051
\__enumext_getkeyans:nn ... 95, 3499, 3503, 3503
\__enumext_getkeyans_aux:n . 95, 3487, 3490, 3490
\l__enumext_hyperref_bool . 24, 27, 28, 129, 237,
    258, 275, 1732, 1942, 3055, 3403
\__enumext_hypertarget:nn 28, 232, 260, 264, 280
\__enumext_if_is_int:n ..... 199
\__enumext_if_is_int:nTF ..... 199, 519, 533
\l__enumext_item_column_pos_vii_int 85, 2768,
    2774, 2780, 2784, 2791, 2983, 3130, 3133
\l__enumext_item_column_pos_viii_int ... 92,
    3192, 3198, 3204, 3208, 3215, 3335, 3464, 3467
\l__enumext_item_column_pos_X_int ..... 153
\g__enumext_item_count_all_vii_int 85, 2792,
    2984, 3141, 3148
\g__enumext_item_count_all_viii_int 92, 3216,
    3336, 3475, 3482
\g__enumext_item_count_all_X_int ..... 153
\__enumext_item_peek_args_vii: 85, 2985, 2987,
    2987
\__enumext_item_peek_args_viii: 92, 3337, 3339,
    3339
\__enumext_item_starred: .. 67, 2147, 2147, 2165
\l__enumext_item_starred_vii_bool 3002, 3018,
    3082
\l__enumext_item_starred_viii_bool 3354, 3370,
    3416
\l__enumext_item_starred_X_bool ..... 153

```


__enumext_item_std:w 27, 65, 66, 80, 219, 223, 2064,
 2070, 2095, 2113, 2117, 2125, 2716
 \g__enumext_item_symbol_aux_vii_tl 3026, 3084,
 3087, 3091, 3093
 \g__enumext_item_symbol_aux_viii_tl .. 3418,
 3421, 3425, 3427
 \g__enumext_item_symbol_aux_X_tl 153
 \l__enumext_item_symbol_sep_vii_dim .. 3035,
 3043, 3090, 3092
 \l__enumext_item_symbol_sep_viii_dim . 3424,
 3426
 \g__enumext_item_symbol_tl 22, 65, 35, 2079, 2153,
 2170
 \l__enumext_item_symbol_vii_tl 3087
 \l__enumext_item_symbol_viii_tl 3421
 \l__enumext_item_text_vii_box 3077, 3118, 3125
 \l__enumext_item_text_viii_box 3411, 3452, 3459
 \l__enumext_item_text_X_box 153
 \l__enumext_item_width_vii_dim ... 2750, 2795,
 2803, 2804
 \l__enumext_item_width_viii_dim .. 3174, 3219,
 3227, 3228
 \l__enumext_item_width_X_dim 153
 \l__enumext_itemindent_X_dim 44
 \l__enumext_itemsep_vii_skip 3147
 \l__enumext_itemsep_viii_skip 3481
 \l__enumext_joined_item_aux_vii_int .. 2789,
 2790, 2791, 2792, 2798
 \l__enumext_joined_item_aux_viii_int . 3213,
 3214, 3215, 3216, 3222
 \l__enumext_joined_item_aux_X_int 153
 __enumext_joined_item_vii:w 85, 86, 2990, 2991,
 2993, 2993
 \l__enumext_joined_item_vii_int .. 2760, 2761,
 2764, 2766, 2772, 2777, 2782, 2787, 2789, 2795
 __enumext_joined_item_viii:w . 92, 3342, 3343,
 3345, 3345
 \l__enumext_joined_item_viii_int . 3184, 3185,
 3188, 3190, 3196, 3201, 3206, 3211, 3213, 3219
 \l__enumext_joined_item_X_int 153
 \l__enumext_joined_width_vii_dim . 2793, 2800,
 2803, 3108, 3120
 \l__enumext_joined_width_viii_dim 3217, 3224,
 3227, 3442, 3454
 \l__enumext_joined_width_X_dim 153
 __enumext_keyans_addto_prop:n 61, 1853, 1853,
 2127, 2648
 __enumext_keyans_addto_seq:n . 62, 1922, 1922,
 2129, 2650
 __enumext_keyans_addto_seq_link: 1922, 1936,
 1938, 3393
 __enumext_keyans_anspic_code:nnn . 78, 2639,
 2642, 2642
 __enumext_keyans_check_ans:nn .. 62, 63, 1960,
 1960, 2524, 2680
 __enumext_keyans_default_item:n .. 66, 2108,
 2108, 2143
 \l__enumext_keyans_env_bool 20, 2380, 2535, 2619
 __enumext_keyans_fake_item: .. 635, 651, 2246
 \l__enumext_keyans_level_h_int 20, 1900, 3301,
 3302
 \l__enumext_keyans_level_int .. 20, 1169, 1655,
 1895, 1990, 2534, 2538, 2633
 __enumext_keyans_make_label: 30, 68, 2173, 2173,
 2245
 __enumext_keyans_mini_addvspace: 45, 76, 1073,
 1073, 2563
 __enumext_keyans_mini_right_cmd:n 47, 1171,
 1194, 1194
 __enumext_keyans_mini_set_vskip: . 44, 1011,
 1011, 1075
 __enumext_keyans_multi_addvspace: . 77, 860,
 871, 2588
 __enumext_keyans_multi_set_vskip: . 41, 860,
 860, 873
 __enumext_keyans_multicols_start: 76, 2567,
 2569, 2569
 __enumext_keyans_multicols_stop: . 77, 1198,
 2594, 2594, 2618
 __enumext_keyans_parse_keys:n 2513, 2547, 2547
 \l__enumext_keyans_pic_above_int . 110, 2726,
 2727, 2729
 \l__enumext_keyans_pic_above_skip .. 80, 110,
 2671, 2710
 __enumext_keyans_pic_arg_two: 80, 2669, 2698,
 2698
 \l__enumext_keyans_pic_below_int . 110, 2726,
 2727, 2730
 \l__enumext_keyans_pic_body_seq .. 78–80, 110,
 2637, 2676, 2734
 __enumext_keyans_pic_do:n 80, 2676, 2678, 2718,
 2718, 2722
 \l__enumext_keyans_pic_level_int .. 20, 1161,
 1659, 1856, 1890, 1925, 2687, 2688
 __enumext_keyans_pic_row:n 80, 2720, 2723, 2723
 __enumext_keyans_pic_safe_exec: .. 79, 2665,
 2685, 2685
 __enumext_keyans_pic_skip_abs:N .. 79, 2693,
 2693, 2709
 \l__enumext_keyans_pic_width_dim . 110, 2725,
 2732
 __enumext_keyans_redefine_item: .. 67, 2131,
 2131, 2244
 __enumext_keyans_safe_exec: . 2512, 2528, 2528
 __enumext_keyans_show_left:n . 66, 1968, 1968,
 2123, 2656
 __enumext_keyans_starred_item:n .. 66, 2120,
 2120, 2139
 __enumext_keyans_store_ref: .. 61, 1869, 1869,
 2128, 2649, 3391
 __enumext_keyans_store_ref_aux_i: 61, 1869,
 1881, 1884
 __enumext_keyans_store_ref_aux_ii: 62, 1869,
 1910, 1912
 \l__enumext_keyans_tmpa_tl .. 23, 82, 2122, 2126
 \l__enumext_label_copy_i_tl .. 1786, 1888, 1893,
 1898, 1903
 \l__enumext_label_copy_v_tl 1898
 \l__enumext_label_copy_vi_tl 1893
 \l__enumext_label_copy_vii_tl 1761, 1772, 1803,
 1888
 \l__enumext_label_copy_viii_tl 1903
 \l__enumext_label_copy_X_tl 131
 \l__enumext_label_fill_left_v_tl 2177
 \l__enumext_label_fill_left_X_tl 70
 \l__enumext_label_fill_right_v_tl 2184
 \l__enumext_label_fill_right_X_tl 70
 \l__enumext_label_font_style_v_tl 2178, 2660

```

\l__enumext_label_font_style_vii_tl ... 3096
\l__enumext_label_font_style_viii_tl .. 3430
\l__enumext_label_i_tl ..... 464
\l__enumext_label_ii_tl ..... 464
\l__enumext_label_iii_tl ..... 464
\l__enumext_label_iv_tl ..... 464
\__enumext_label_style:Nnn 22, 29, 316, 316, 331,
    469, 489, 505, 509
\l__enumext_label_v_tl .. 61, 62, 502, 1861, 1930,
    1972, 1979, 1995, 2002, 2122, 2126, 2516, 2655, 2657
\l__enumext_label_vi_tl . 61, 62, 502, 1858, 1927,
    2655, 2657, 2661
\l__enumext_label_vii_tl . 484, 3013, 3038, 3045
\l__enumext_label_viii_tl 484, 3365, 3388, 3389
\l__enumext_label_width_by_box .. 40, 312, 313
\__enumext_label_width_by_box:Nn 29, 310, 310,
    315, 327, 543
\l__enumext_labelsep_i_dim ..... 1976, 1999
\l__enumext_labelsep_v_dim ..... 2578
\l__enumext_labelsep_vii_dim . 2745, 2754, 2796,
    3036, 3106, 3122
\l__enumext_labelsep_viii_dim 3169, 3178, 3220,
    3440, 3456
\l__enumext_labelwidth_i_dim ..... 1975, 1998
\l__enumext_labelwidth_v_dim ..... 2578
\l__enumext_labelwidth_vii_dim ... 2745, 2753,
    2796, 3099, 3103, 3121
\l__enumext_labelwidth_viii_dim .. 3169, 3177,
    3220, 3433, 3437, 3455
\l__enumext_leftmargin_tmp_v_bool . 80, 2700
\l__enumext_leftmargin_tmp_X_bool ..... 44
\l__enumext_leftmargin_tmp_X_dim ..... 44
\l__enumext_leftmargin_X_dim ..... 44
\__enumext_level: 186, 186, 192, 193, 197, 407, 409,
    410, 418, 420, 638, 642, 646, 713, 717, 721, 725, 809,
    811, 813, 815, 848, 850, 852, 854, 858, 898, 901, 920,
    929, 935, 940, 944, 955, 959, 960, 965, 1001, 1005,
    1178, 1184, 1231, 1233, 1235, 1238, 1245, 1247, 1249,
    1252, 1429, 1430, 1432, 1563, 1571, 1575, 1579, 1824,
    1827, 1828, 2061, 2063, 2064, 2068, 2069, 2070, 2077,
    2079, 2083, 2084, 2087, 2092, 2094, 2095, 2149, 2152,
    2154, 2161, 2162, 2163, 2166, 2169, 2311, 2313, 2349,
    2354, 2355, 2356, 2358, 2362, 2367, 2368, 2369, 2371,
    2384, 2389, 2396, 2407, 2409, 2412, 2413, 2415, 2420,
    2427, 2430, 2432, 2434, 2435, 2436, 2437, 2440, 2446,
    2451, 2457, 2460, 2462, 2475, 3066, 3067
\__enumext_level_ ..... 192
\__enumext_level_end:n ..... 190, 195
\l__enumext_level_h_int 20, 426, 450, 1780, 1797,
    2332, 2382, 2879, 2880
\l__enumext_level_int 20, 188, 821, 972, 1165, 1408,
    1757, 1767, 1773, 1779, 1787, 1795, 1802, 2259, 2325,
    2326, 2341, 2387, 2442, 2502, 2542, 2629, 2888, 2964,
    2974, 3155, 3308
\__enumext_level_set:n ..... 190, 190
\__enumext_list_arg_two_i: ..... 2219
\__enumext_list_arg_two_ii: ..... 2219
\__enumext_list_arg_two_iii: ..... 2219
\__enumext_list_arg_two_iv: ..... 2219
\__enumext_list_arg_two_v: . 67, 2219, 2518, 2701
\__enumext_list_arg_two_vii: ..... 2265, 2862
\__enumext_list_arg_two_viii: ..... 2265, 3285
\l__enumext_listoffset_v_dim ..... 2580
\l__enumext_listparindent_vii_dim .... 3109
\l__enumext_listparindent_viii_dim ... 3443
\__enumext_make_label: 30, 65, 67, 2157, 2157, 2254
\l__enumext_mark_answer_sym_tl ... 56, 63, 105,
    1477, 1633, 1839, 1983
\l__enumext_mark_position_str 105, 1481, 1482,
    1504, 1505, 1631
\l__enumext_mark_ref_sym_tl .. 105, 1491, 1737,
    1950
\__enumext_mini_addvspace: 43, 73, 994, 994, 2417
\__enumext_mini_addvspace_vii: 46, 1147, 1147,
    2821
\__enumext_mini_addvspace_viii: 46, 1147, 1153,
    3245
__enumext_mini_env* ..... 888
\__enumext_mini_right_cmd:n 47, 1173, 1175, 1175
\__enumext_mini_set_vskip: ... 42, 895, 895, 996
\__enumext_mini_set_vskip_vii: 45, 1090, 1090,
    1149
\__enumext_mini_set_vskip_viii: 45, 1090, 1112,
    1155
\__enumext_minipage:w 27, 225, 227, 890, 2732, 3108,
    3442
\l__enumext_minipage_active_v_bool ... 76, 77,
    2561, 2586, 2599, 2607
\g__enumext_minipage_active_vii_bool ... 82,
    2832, 2837, 2849
\l__enumext_minipage_active_vii_bool . 2817,
    2828
\g__enumext_minipage_active_viii_bool 3256,
    3261, 3273
\l__enumext_minipage_active_viii_bool 3241,
    3252
\g__enumext_minipage_active_X_bool ... 153
\l__enumext_minipage_active_X_bool .... 58
\g__enumext_minipage_after_skip 58, 1094, 1106,
    2847, 3271
\l__enumext_minipage_after_skip 42, 43, 74, 77,
    58, 911, 926, 946, 962, 977, 983, 989, 1003, 1013, 1022,
    1025, 1037, 1055, 1066, 1082, 1114, 1127, 1141, 2484,
    2616
\g__enumext_minipage_center_vii_bool . 2841,
    2850
\g__enumext_minipage_center_viii_bool 3265,
    3274
\g__enumext_minipage_center_X_bool ... 153
\l__enumext_minipage_hsep_v_dim ... 76, 2559
\l__enumext_minipage_hsep_vii_dim .... 2815
\l__enumext_minipage_hsep_viii_dim ... 3239
\l__enumext_minipage_left_skip 42, 76, 58, 903,
    918, 937, 952, 999, 1009, 1014, 1020, 1029, 1046, 1058,
    1078, 1088, 1092, 1097, 1101, 1115, 1119, 1133, 1151,
    1157
\l__enumext_minipage_left_v_dim 76, 2557, 2565
\l__enumext_minipage_left_vii_dim 2811, 2823
\l__enumext_minipage_left_viii_dim 3235, 3247
\l__enumext_minipage_left_X_dim ..... 58
\g__enumext_minipage_right_skip 58, 1093, 1098,
    1102, 2840, 3264
\l__enumext_minipage_right_skip .. 42, 58, 907,
    922, 942, 957, 1015, 1021, 1033, 1051, 1062, 1116,
    1123, 1137, 1185, 1202
\l__enumext_minipage_right_v_dim .. 76, 1196,
    1201, 2555, 2559
\g__enumext_minipage_right_vii_dim 82, 2819,
    2839, 2852

```

`\l__enumext_minipage_right_vii_dim` 82, 2809, 2814, 2820
`\g__enumext_minipage_right_viii_dim` .. 3243, 3263, 3276
`\l__enumext_minipage_right_viii_dim` .. 3233, 3238, 3244
`\g__enumext_minipage_right_X_dim` 153
`\g__enumext_minipage_right_X_skip` 153
`\g__enumext_minipage_stat_int` . 73, 76, 58, 1190, 1207, 2416, 2477, 2482, 2562, 2609, 2614
`\g__enumext_miniright_code_vii_tl` . 82, 2845, 2851
`\g__enumext_miniright_code_viii_tl` 3269, 3275
`\g__enumext_miniright_code_X_tl` 153
`__enumext_multi_addvspace:` ... 40, 74, 843, 843, 2448
`__enumext_multi_set_vskip:` .. 40, 807, 807, 845
`\l__enumext_multicols_above_ii_skip` ... 826
`\l__enumext_multicols_above_iii_skip` .. 832
`\l__enumext_multicols_above_iv_skip` ... 838
`\l__enumext_multicols_above_v_skip` 862, 876, 886
`\l__enumext_multicols_above_X_skip` 52
`\l__enumext_multicols_below_v_skip` 866, 880, 2601
`\l__enumext_multicols_below_X_skip` 52
`__enumext_multicols_start:` 73, 2422, 2424, 2424
`__enumext_multicols_stop:` 74, 1180, 2454, 2454, 2486
`__enumext_newlabel:nn` 24, 28, 60, 268, 268, 1813, 1916
`\l__enumext_newlabel_arg_one_tl` 24, 28, 60, 61, 131, 1736, 1806, 1814, 1905, 1917, 1948
`\l__enumext_newlabel_arg_two_tl` 24, 28, 59, 131, 1760, 1770, 1784, 1800, 1815, 1892, 1897, 1902, 1918
`__enumext_parse_keys:n` 2307, 2338, 2338
`__enumext_parse_keys_vii:n` .. 2857, 2894, 2894
`__enumext_parse_keys_viii:n` . 3281, 3313, 3313
`__enumext_parse_store_keys:n` 71, 72, 2344, 2347, 2347
`__enumext_parse_store_keys_vii:n` . 84, 2901, 2905, 2905
`\l__enumext_parsep_i_skip` . 824, 826, 975, 1023
`\l__enumext_parsep_ii_skip` 830, 832, 981
`\l__enumext_parsep_iii_skip` 836, 838, 987
`\l__enumext_parsep_vii_skip` 3110
`\l__enumext_parsep_viii_skip` 3444
`\l__enumext_partopsep_v_skip` .. 878, 882, 1049, 1053, 1060, 1064, 1080, 1084
`\l__enumext_partopsep_viii_skip` 1125
`__enumext_phantomsection:` 28, 232, 261, 265, 281
`__enumext_print_footnote:` ... 2021, 2044, 3128, 3462
`__enumext_print_keyans_box:NN` 56, 1625, 1625, 1638, 1826, 1974, 1997
`\l__enumext_print_keyans_i_tl` 3513, 3542
`\l__enumext_print_keyans_ii_tl` ... 3518, 3543
`\l__enumext_print_keyans_iii_tl` .. 3523, 3544
`\l__enumext_print_keyans_iv_tl` ... 3528, 3545
`\l__enumext_print_keyans_vii_tl` .. 3533, 3546
`\l__enumext_print_keyans_X_tl` 94
`__enumext_printkeyans:nnn` . 96, 3547, 3550, 3550
`__enumext_redefine_item:` . 66, 2097, 2097, 2253
`\l__enumext_ref_aux_tl` 142, 407, 409, 412, 428, 430, 433, 438, 440, 443
`\l__enumext_ref_key_arg_tl` .. 142, 401, 406, 413, 425, 434, 444
`__enumext_regex_label_ref_key:` . 31, 396, 396, 408, 429, 439
`__enumext_register_counter_style:Nn` .. 300, 300, 305, 306, 307, 308, 309
`__enumext_remove_extra_parsep_vii:` .. 2872, 3137, 3137
`__enumext_remove_extra_parsep_viii:` . 3295, 3471, 3471
`__enumext_renew_footnote:` ... 2021, 2025, 3080, 3414
`\l__enumext_resume_bool` 22, 35, 1360, 2236
`__enumext_resume_counter:` . 50, 1326, 1351, 1351
`__enumext_resume_counter_star:` 1328
`__enumext_resume_counter_vii:` 50, 1335, 1351, 1362
`\g__enumext_resume_int` 22, 75, 35, 1355, 1366, 2237, 2490
`\l__enumext_resume_vii_bool` ... 35, 1371, 2283
`\g__enumext_resume_vii_int` .. 84, 35, 2284, 2945
`__enumext_safe_exec:` 2306, 2323, 2323
`__enumext_safe_exec_vii:` ... 2856, 2877, 2877
`__enumext_safe_exec_viii:` ... 3280, 3299, 3299
`__enumext_set_error:nn` 3628, 3638, 3640
`__enumext_set_label_ref:n` ... 31, 404, 404, 476
`__enumext_set_label_ref_h:n` . 31, 423, 423, 496
`__enumext_set_parse:n` 3611, 3628, 3628
`\l__enumext_setkey_tmpa_int` ... 89, 3604, 3608
`\l__enumext_setkey_tmpa_seq` 89, 3602, 3612, 3618, 3620, 3622, 3635
`\l__enumext_setkey_tmpa_tl` 89, 3610, 3614
`\l__enumext_setkey_tmpb_seq` 89, 3603, 3606, 3610, 3611
`\l__enumext_setkey_tmpb_tl` 89, 3630, 3632, 3633
`\l__enumext_show_answer_bool` . 105, 1485, 1489, 1508, 1512, 1833, 1970, 2652
`__enumext_show_length:nnn` .. 36, 205, 205, 3680, 3681, 3682, 3683, 3684, 3685, 3686, 3687, 3688, 3689, 3695, 3696, 3697, 3698, 3699, 3700, 3701, 3702, 3703, 3704
`\l__enumext_show_position_bool` 105, 1486, 1488, 1509, 1511, 1837, 1981, 2653
`\g__enumext_standar_bool` 20, 2335
`\l__enumext_standar_bool` . 20, 1765, 1778, 1794, 2328, 2489, 2887
`__enumext_standard_item_vii:w` 86, 2998, 3000, 3000
`__enumext_standard_item_viii:w` . 92, 93, 3350, 3352, 3352
`\g__enumext_starred_bool` . 83, 84, 20, 1407, 1756, 1766, 1796, 1886, 2467, 2891, 2952, 2957
`\l__enumext_starred_bool` . 83, 84, 20, 1689, 1697, 1781, 1822, 2331, 2884, 2958
`__enumext_starred_columns_set_vii:` .. 2739, 2739, 2865
`__enumext_starred_columns_set_viii:` . 3163, 3163, 3288
`__enumext_starred_item:nn` ... 2073, 2073, 2103
`__enumext_starred_item_vii:w` . 86, 2997, 3016, 3016

```

\__enumext_starred_item_vii_aux_i:w .. 3016,
    3021, 3024
\__enumext_starred_item_vii_aux_ii:w . 3016,
    3022, 3027, 3029
\__enumext_starred_item_vii_aux_iii:w 3016,
    3032, 3041
\__enumext_starred_item_viii:w .. 92, 93, 3349,
    3368, 3368
\__enumext_starred_item_viii_aux_i:w . 3368,
    3373, 3376
\__enumext_starred_item_viii_aux_ii:w 3368,
    3374, 3383, 3385
\__enumext_starred_joined_item_vii:n . 81, 86,
    2758, 2758, 2995
\__enumext_starred_joined_item_viii:n 89, 92,
    3182, 3182, 3347
\__enumext_start_from:NNn 33, 517, 517, 530, 552
\__enumext_start_item_tmp_vii: 83, 2868, 2980,
    2980
\__enumext_start_item_tmp_viii: 91, 3291, 3332,
    3332
\__enumext_start_item_vii:w . 86, 87, 3008, 3013,
    3038, 3045, 3047, 3047
\__enumext_start_item_viii:w .. 93, 3360, 3365,
    3388, 3395, 3395
\__enumext_start_list:nn 27, 69, 80, 219, 221, 2310,
    2515, 2666, 2860, 3283
\__enumext_start_mini_vii: . 84, 2807, 2807, 2938
\__enumext_start_mini_viii: 92, 3231, 3231, 3324
\__enumext_start_store_level: . 72, 2309, 2376,
    2376
\__enumext_start_store_level_vii: . 85, 2859,
    2960, 2960
\l__enumext_start_X_int ..... 70, 547
\__enumext_stop_item_tmp_vii: . 83, 85, 87, 2867,
    2871, 2982, 3049
\__enumext_stop_item_tmp_viii: .. 91-93, 3290,
    3294, 3334, 3397
\__enumext_stop_item_vii: 87, 88, 3049, 3113, 3113
\__enumext_stop_item_viii: .. 93, 94, 3397, 3447,
    3447
\__enumext_stop_list: .. 27, 219, 222, 2319, 2525,
    2679, 2873, 3296
\__enumext_stop_mini_vii: 82, 84, 2826, 2826, 2942
\__enumext_stop_mini_viii: . 92, 3231, 3250, 3328
\__enumext_stop_store_level: .. 72, 2320, 2376,
    2394
\__enumext_stop_store_level_vii: .. 85, 2874,
    2960, 2970
\l__enumext_store_active_bool 23, 50, 71, 84, 82,
    1341, 1353, 1364, 1651, 2342, 2379, 2530, 2537, 2625,
    2683, 2899, 2962, 2972, 3307
\__enumext_store_addto_prop:n 55, 61, 1542, 1542,
    1551, 1676, 1867, 3390
\__enumext_store_addto_seq:n 55, 62, 1552, 1552,
    1558, 1565, 1579, 1587, 1596, 1614, 1622, 1740, 1953
\l__enumext_store_ans_bool 115, 1380, 1427, 1561,
    1585, 1592, 1620, 1664, 3064
\l__enumext_store_anskey_arg_tl .. 23, 58, 82,
    1682, 1691, 1693, 1699, 1707, 1710, 1720, 1725, 1728,
    1734, 1740
\__enumext_store_anskey_code:nnnn . 57, 1670,
    1674, 1674
\__enumext_store_anskey_show_left:n 60, 1681,
    1831, 1831
\__enumext_store_anskey_show_wrap:n 60, 1819,
    1819, 1835, 1850
\l__enumext_store_columns_break_bool . 1645,
    1688
\l__enumext_store_columns_join_int 82, 1696,
    1701
\l__enumext_store_columns_sep_vii_bool 2920
\l__enumext_store_columns_sep_vii_dim 2925,
    2929
\l__enumext_store_columns_sep_X_bool ... 94
l__enumext_store_columns_sep_X_dim .... 94
\l__enumext_store_columns_vii_bool ... 2907
\l__enumext_store_columns_vii_int 2912, 2916
\l__enumext_store_columns_X_bool ..... 94
\l__enumext_store_columns_X_int ..... 94
\__enumext_store_internal_ref: .. 57, 59, 1679,
    1742, 1742
\l__enumext_store_item_symbol_sep_dim 1643,
    1717, 1722
\l__enumext_store_item_symbol_tl . 1641, 1708,
    1712
\l__enumext_store_keyans_label_tl 23, 61, 62,
    82, 1855, 1858, 1861, 1865, 1867, 1924, 1927, 1930,
    1934, 1944, 1953, 1954, 3378, 3381, 3389, 3390, 3392
\__enumext_store_level_close: . 55, 1559, 1583,
    2398
\__enumext_store_level_close_vii: 1590, 1618,
    2976
\__enumext_store_level_open: .. 54, 55, 72, 1559,
    1559, 2385, 2390
\__enumext_store_level_open_vii: .. 84, 1590,
    1590, 2966
\g__enumext_store_name_tl 23, 74, 82, 1445, 1448,
    1463, 1466, 2470, 2508, 2955, 3161
\l__enumext_store_name_tl 23, 50, 82, 1340, 1357,
    1368, 1544, 1545, 1546, 1548, 1554, 1555, 1556, 1808,
    1809, 1845, 1907, 1908, 1989, 2470, 2491, 2494, 2946,
    2949, 2955
\l__enumext_store_opt_vii_tl . 1594, 1604, 1610,
    1614, 2914, 2927
\l__enumext_store_opt_X_tl ..... 94
\l__enumext_store_ref_key_bool 57, 1494, 1677,
    1731, 1871, 1941
\l__enumext_store_upper_level_X_bool ... 94
\l__enumext_store_write_aux_file_tl 24, 60, 62,
    131, 1811, 1817, 1914, 1920
\__enumext_storing_set:n . 50, 1324, 1333, 1338,
    1338
\l__enumext_the_counter_vii_tl ..... 430
\l__enumext_the_counter_viii_tl ..... 440
\l__enumext_the_counter_X_tl ..... 142
\__enumext_tmp:n 30, 34, 44, 51, 52, 57, 64, 69, 70, 81,
    94, 104, 123, 128, 134, 138, 146, 152, 153, 172, 213,
    217, 630, 634, 1373, 1390, 1470, 1498, 1499, 1516,
    1744, 1751, 1752, 1773, 1787, 1790, 1802, 1873, 1880,
    2219, 2264, 2265, 2303
\__enumext_tmp:nn 332, 353, 354, 382, 383, 395, 547,
    566, 611, 629, 687, 695, 696, 710, 775, 791, 792, 806,
    1211, 1227, 1517, 1541, 2005, 2020
\__enumext_tmp:nnn 464, 480, 481, 482, 483, 484, 500,
    501
\__enumext_tmp:nnnnnn 567, 592, 595, 598, 600, 602,
    605, 608

```

<code>_enumext_tmp:w</code>	3496, 3499
<code>\l_enumext_tmpa_vii_int</code>	2749, 2752
<code>\l_enumext_tmpa_viii_int</code>	3173, 3176
<code>\l_enumext_tmpa_X_int</code>	153
<code>\l_enumext_topsep_v_skip</code> 864, 868, 1018, 1031, 1039, 1044, 1064, 1068, 2682, 2713	
<code>\l_enumext_topsep_vii_skip</code> ..	1095, 1104, 1108
<code>\l_enumext_topsep_viii_skip</code> ..	1117, 1139, 1143
<code>_enumext_use_key_ref:</code>	31, 416, 416, 2255
<code>_enumext_use_key_ref_h:</code> ..	32, 448, 448, 2289
<code>\l_enumext_vspace_a_star_v_bool</code>	1260
<code>\l_enumext_vspace_a_star_vii_bool</code> ...	1282
<code>\l_enumext_vspace_a_star_viii_bool</code> ...	1293
<code>\l_enumext_vspace_a_star_X_bool</code>	70
<code>_enumext_vspace_above:</code> ..	48, 1228, 1228, 2403
<code>_enumext_vspace_above_v:</code> ..	49, 1256, 1256, 2553
<code>\l_enumext_vspace_above_v_skip</code> ..	1258, 1262, 1264
<code>_enumext_vspace_above_vii:</code> ..	49, 1278, 1278, 2936
<code>\l_enumext_vspace_above_vii_skip</code> ..	1280, 1284, 1286
<code>_enumext_vspace_above_viii:</code> ..	49, 1278, 1289, 3322
<code>\l_enumext_vspace_above_viii_skip</code> ..	1291, 1295, 1297
<code>\l_enumext_vspace_b_star_v_bool</code>	1271
<code>\l_enumext_vspace_b_star_vii_bool</code> ...	1304
<code>\l_enumext_vspace_b_star_viii_bool</code> ...	1315
<code>\l_enumext_vspace_b_star_X_bool</code>	70
<code>_enumext_vspace_below:</code> ..	48, 1242, 1242, 2488
<code>_enumext_vspace_below_v:</code> ..	49, 1267, 1267, 2621
<code>\l_enumext_vspace_below_v_skip</code> ..	1269, 1273, 1275
<code>_enumext_vspace_below_vii:</code> ..	49, 1300, 1300, 2944
<code>\l_enumext_vspace_below_vii_skip</code> ..	1302, 1306, 1308
<code>_enumext_vspace_below_viii:</code> ..	49, 1300, 1311, 3330
<code>\l_enumext_vspace_below_viii_skip</code> ..	1313, 1317, 1319
<code>_enumext_widest_from:nNNn</code> ..	34, 531, 531, 546, 558
<code>\g_enumext_widest_label_tl</code> 22, 29, 40, 320, 324, 328	
<code>\l_enumext_wrap_label_opt_v_bool</code>	2116
<code>\l_enumext_wrap_label_opt_vii_bool</code> ..	86, 3007
<code>\l_enumext_wrap_label_opt_viii_bool</code> ..	93, 3359
<code>\l_enumext_wrap_label_opt_X_bool</code>	70
<code>\l_enumext_wrap_label_v_bool</code> 2112, 2116, 2124, 2179	
<code>\l_enumext_wrap_label_vii_bool</code> ..	86, 3006, 3011, 3019, 3097
<code>\l_enumext_wrap_label_viii_bool</code> ..	93, 3358, 3363, 3371, 3431
<code>\l_enumext_wrap_label_X_bool</code>	70
<code>_enumext_wrapper_label_v:n</code>	2181, 2661
<code>_enumext_wrapper_label_vii:n</code>	3100
<code>_enumext_wrapper_label_viii:n</code>	3434
<code>_enumext_zero_count_level:</code> ..	211, 211, 1450, 1468
<code>_enumext_zero_parsep:</code>	43, 915, 970, 970
<code>enumext*</code>	4, 2854
<code>enumXi</code>	292
<code>enumXii</code>	292
<code>enumXiii</code>	292
<code>enumXiv</code>	292
<code>enumXv</code>	292
<code>enumXvi</code>	292
<code>enumXvii</code>	292
<code>enumXviii</code>	292
Environments provide by <code>enumext</code> :	
<code>enumext*</code> ..	21, 22, 24–26, 29–33, 35, 36, 38, 39, 45, 46, 49–51, 53–59, 61, 64, 71, 72, 83–85, 87, 89, 90, 92, 96, 99, 100
<code>enumext</code> ..	21, 22, 24, 26, 29–32, 34–37, 39–48, 50–59, 61, 64–67, 69, 71, 72, 75, 79, 81, 82, 85, 96, 99
<code>keyans*</code> ..	21–23, 25, 26, 29–33, 35, 36, 38, 39, 45, 46, 49, 51, 54, 55, 61, 64, 91, 93, 99, 100
<code>keyanspic</code> ..	21–24, 29, 30, 33, 46, 50, 51, 55, 61–63, 78–80, 100
<code>keyans</code> ..	21–24, 26, 29, 30, 33–39, 41, 44–51, 54, 55, 61–63, 67–69, 75, 76, 78, 80, 82, 92, 99, 100
Environments:	
<code>enumext*</code>	70
<code>keyans*</code>	70
<code>list</code>	26, 27, 68, 69, 71
<code>lrbox</code>	81, 87, 88, 94
<code>minipage</code>	26, 27, 39, 41, 78–81, 87, 88, 94
<code>multicols</code>	39–42, 47, 73, 74, 76, 77
exp commands:	
<code>\exp_after:wN</code>	3499
<code>\exp_args:Ne</code>	2340, 3487
<code>\exp_not:N</code> ..	150, 323, 412, 433, 443, 644, 658, 659, 670, 671, 682, 683, 1736, 1842, 1843, 1946, 1986, 1987, 3496
<code>\exp_not:n</code> ..	412, 413, 433, 434, 443, 444, 645, 1525, 1532, 1701, 1712, 1722, 1736, 1737, 1814, 1917, 1948, 1950, 2358, 2371, 2916, 2929
F	
<code>\fbox</code>	1475
file commands:	
<code>\file_input_stop:</code>	3797
<code>first</code>	696
<code>font</code>	332
<code>\footnote</code>	64
<code>\footnote</code>	64, 2029
<code>\footnotemark</code>	2039
<code>\footnotesize</code>	1843, 1987
<code>\footnotetext</code>	2023
G	
<code>\getkeyans</code>	13, 95, 3485
group commands:	
<code>\group_begin:</code> ..	1663, 1841, 1985, 3076, 3095, 3410, 3429, 3507, 3541
<code>\group_end:</code> ..	1672, 1848, 1993, 3105, 3117, 3439, 3451, 3509, 3548
H	
<code>\hbadness</code>	3124, 3458
hbox commands:	
<code>\hbox_set:Nn</code>	312
<code>\hfill</code> ..	362, 366, 371, 372, 1182, 1200, 1736, 1946, 2831, 3255
hook commands:	
<code>\hook_gput_code:nnn</code>	9, 180, 184, 230
<code>\hook_gset_rule:nnnn</code>	231

\hspace	3135, 3469
\hyperlink	58, 62
\hyperlink	1736, 1946
\hypertarget	28
\hypertarget	260
I	
\IfHyperBoolean	238
\IfPackageLoadedTF	11, 234, 248
\ignorespaces	647
int commands:	
\int_add:Nn	2791, 3215
\int_case:nn	821, 972
\int_compare:nNnTF	426, 450, 897, 1016, 1161, 1165, 1169, 1442, 1460, 1655, 1659, 1856, 1890, 1895, 1900, 1925, 2326, 2382, 2387, 2426, 2442, 2456, 2477, 2502, 2538, 2542, 2571, 2596, 2609, 2629, 2633, 2688, 2761, 2771, 2787, 2880, 2964, 2974, 3130, 3139, 3155, 3185, 3195, 3211, 3302, 3308, 3464, 3473, 3608
\int_compare_p:nNn	1408, 1757, 1767, 1779, 1780, 1795, 1797, 2332, 2888
\int_decr:N	2790, 3214
\int_eval:n	1548, 1809, 1843, 1908, 1987, 2237, 2240, 2284, 2287, 2779, 3203
\int_from_alph:n	525, 539
\int_from_roman:n	527, 541
\int_gadd:Nn	1429, 2792, 3066, 3216
\int_gincr:N	1432, 1668, 1957, 2060, 2061, 2091, 2092, 2416, 2562, 2984, 3070, 3071, 3336
\int_gset:Nn	1355, 1366, 2037
\int_gset_eq:NN	1411, 1414, 1416, 1417, 1418, 1419, 1420, 1421, 2034, 2490, 2493, 2945, 2948
\int_gzero:N	215, 1190, 1207, 2482, 2614, 3148, 3482
\int_if_exist:NTF	1342, 1393, 1395, 1397, 1399, 1401, 1403, 2491, 2946
\int_incr:N	2325, 2534, 2687, 2879, 2983, 3301, 3335
\int_mod:nn	3141, 3475
\int_new:N	20, 21, 22, 23, 24, 36, 38, 58, 74, 86, 91, 99, 112, 113, 120, 121, 122, 125, 126, 139, 156, 157, 158, 159, 160, 1344, 1394, 1396, 1398, 1400, 1402, 1404
\int_set:Nn	521, 525, 527, 1437, 1454, 1522, 1696, 2726, 2727, 2749, 2760, 2766, 2782, 3124, 3173, 3184, 3190, 3206, 3458, 3604
\int_set_eq:NN	2353, 2789, 2911, 3213
\int_step_function:nnN	1773, 1787, 1802
\int_step_inline:nnn	2728, 3631
\int_to_roman:n	188, 1753, 1791
\int_use:N	898, 1357, 1368, 1430, 2240, 2259, 2287, 2341, 2427, 2436, 2451, 2457, 2764, 2765, 2777, 3067, 3188, 3189, 3201
\int_zero:N	3133, 3467
\c_one_int	2749, 2768, 2774, 2780, 2784, 2787, 3173, 3192, 3198, 3204, 3208, 3211
\c_zero_int	1408, 1757, 1767, 1779, 1780, 1795, 1797, 2332, 2888, 2964, 2974, 3144, 3478
\item	27, 37, 38, 56, 65, 78, 80, 81, 83, 91
\item	65, 66, 85, 87, 92, 93, 223, 1567, 1573, 1598, 1606, 1693, 1927, 1930, 2099, 2133, 2866, 2868, 3289, 3291, 3392
\item*	5, 11, 2131
item-pos*	2005
item-sym*	2005
\itemindent	22, 69
\itemindent	68
itemindent	611
\itemsep	79, 80

\itemsep	2702, 2708
\itemwidth	2756, 2800, 2804, 3180, 3224, 3228

K

keyans	11, 2510
keyans*	11, 3278
keyanspic	12, 2663

Keys for environments provide by **enumext**:

above*	23, 48, 49
above	23, 48, 49, 72, 76, 84, 92
after	37, 38, 74, 77, 84, 92
align	23, 30, 67, 87
before*	37, 38, 72, 84, 92
before	37, 38, 76
below*	23, 48, 49
below	23, 48, 49, 74, 77, 84, 92
check-ans	23, 24, 26, 50-52, 57, 62, 63, 65, 66, 73-75, 89, 99
columns-sep*	24, 54, 72, 84
columns-sep	39, 55, 72, 73, 76, 84
columns*	24, 54, 72, 84
columns	22, 39, 42, 48, 55, 72, 73, 76, 84
first	37, 38, 87
font	30, 67, 87
item-pos*	57, 58, 64
item-sym*	22, 57, 58, 64, 65
item*-sep	65
itemindent	23, 35, 36, 67, 87
itemsep	34, 70
labelsep	30, 65, 69, 87
labelwidth	29, 30, 32-34, 69
label	22, 29, 33, 34, 81
lisparindent	70
list-indent	22, 35, 80
list-offset	35
listparindent	35, 87
mark-ans	24, 53, 60
mark-pos	53, 54
mark-ref	24, 53, 59
mini-env	23, 39, 41, 47, 48, 64, 73, 76, 82, 84, 90, 92
mini-sep	23, 39, 73, 76
miniright*	23, 39
miniright	23, 39, 46, 82
minirigth*	26
minirigth	26
no-store	24, 51, 52
noitemsep	34, 43
nosep	34, 43
parindent	70
parsep	34, 70, 87
partopsep	34
ref	25, 31
resume	22, 50, 69, 75, 84, 85
rightmargin	35
save-ans	23, 50, 55, 57, 61, 62, 66, 75, 78, 85, 92, 95, 96
save-key	24
show-ans	24, 53, 54, 56, 57, 60, 66
show-length	26, 36, 69, 99
show-pos	24, 53, 54, 56, 57, 60, 66
start	23, 26, 33, 34, 69
store-brk	57, 58
store-ref	24, 28, 53, 57-59, 61, 62, 67
topsep	34
widest	22, 26, 34
wrap-ans	53, 56, 60

wrap-label*	30, 65, 67, 86, 87, 93
wrap-label	30, 67, 86, 87, 93
keys commands:	
\keys_define:nn	334, 356, 385, 466, 486, 502, 549, 569, 613, 632, 689, 698, 777, 794, 1213, 1322, 1331, 1375, 1472, 1501, 1519, 1639, 2007, 3511, 3574
\l_keys_key_str	3665
\keys_set:nn	348, 801, 1218, 1223, 1685, 2340, 2549, 2898, 3317, 3576, 3577, 3578, 3579, 3580, 3581, 3582, 3583, 3584, 3585, 3586, 3587, 3625
L	
label	464, 484, 502
Labels provide by enumext:	
\Alph*	29
\Roman*	29
\alph*	29
\arabic*	29, 31
\roman*	29
\labelsep	80
\labelsep	2703, 2706
labelsep	332
\labelwidth	29, 80
\labelwidth	2703, 2704
labelwidth	332
\leftmargin	22, 69
\leftmargin	68, 2703
legacy commands:	
\legacy_if:nTF	3050, 3053, 3398, 3401
\legacy_if_gset_false:n	891
\legacy_if_set_false:n	3052, 3400
\legacy_if_set_true:n	3012, 3037, 3044, 3057, 3364, 3387, 3405
\linewidth	73, 76
\linewidth	2411, 2559, 2725, 2752, 2813, 3176, 3237
\list	27
\list	221
list-indent	611
list-offset	611
\listparindent	2705
listparindent	611
\lrbox	3077, 3411
M	
\makebox	81
\makebox	1629, 1631, 2153, 3091, 3099, 3103, 3425, 3433, 3437
\makelabel	65, 67, 68, 81
\makelabel	67, 68, 2159, 2175
\makesavenoteenv	254
mark-ans	1470
mark-pos	1470, 1499
mark-ref	1470
mini-env	775
mini-sep	775
\minipage	27
\minipage	227
\miniright	9, 46, 1159, 2480, 2612
\miniright*	9
mode commands:	
\mode_if_vertical:TF	846, 874, 997, 1076
\mode_leave_vertical:	644, 658, 670, 682, 1598, 1606, 1627, 2151, 3089, 3423
msg commands:	
\msg_error:nn	2540, 2544, 2631, 2690, 2882, 3304, 3310, 3588

\msg_error:nnn	1163, 1167, 1192, 1209, 3501, 3506, 3571, 3641
\msg_error:nnnn	1653, 1657, 1661, 2532, 2627, 2635
\msg_fatal:nn	2327
\msg_fatal:nnn	286
\msg_info:nnn	13, 16, 236, 250
\msg_line_context:	3669, 3674, 3679, 3694, 3717, 3721, 3725, 3730, 3735, 3740, 3745, 3749, 3754, 3759, 3763, 3768, 3772, 3777, 3782, 3787, 3791, 3795
\msg_new:nnn	3642, 3646, 3650, 3654, 3659, 3663, 3667, 3672, 3677, 3692, 3707, 3714, 3719, 3723, 3728, 3733, 3738, 3743, 3747, 3752, 3757, 3761, 3766, 3770, 3775, 3780, 3785, 3789, 3793
\msg_term:nnn	1445, 1463
\msg_term:nnnn	2249, 2259, 2294, 2299
\msg_warning:nn	2479, 2611
\msg_warning:nnn	1448, 1466
\msg_warning:nnnn	1964, 2191, 2196, 2763, 2776, 3187, 3200
\multicolsep	73, 77
\multicolsep	2441, 2584
N	
\NeedsTeXFormat	3
\newcounter	289
\NewDocumentCommand	1159, 1649, 2623, 3485, 3539, 3595
\NewDocumentEnvironment	2304, 2510, 2663, 2854, 3278
\newlabel	28
\newlabel	272
no-store	1373
\noindent	83, 91
\noindent	2418, 2564, 2822, 2867, 3132, 3246, 3290, 3466
\nointerlineskip	2418, 2564, 2822, 3246
noitemsep	567
\nopagebreak	857, 885, 1008, 1087, 1150, 1156
\normalfont	1842, 1986
nosep	567
P	
Packages:	
enumext	21, 50, 68, 78, 98
enumitem	28, 29
expl3	81
footnotehyper	27
hyperref	24, 26-28, 31, 58, 62, 87, 93, 94, 98
lua-visual-debug	41
multicol	21, 98
shortlst	81
\par	857, 885, 1008, 1087, 1150, 1156, 1185, 1202, 1821, 2462, 2484, 2601, 2616, 2737, 2840, 2847, 3132, 3146, 3264, 3271, 3466, 3480
\parindent	3109, 3443
\parsep	40, 43, 79, 80
\parsep	1599, 1607, 2279, 2702, 2709, 2714
parsep	567
\parskip	3110, 3444
\partopsep	80
\partopsep	2280, 2707
partopsep	567
peek commands:	
\peek_meaning:NTF	2989, 3003, 3020, 3031, 3341, 3355, 3372
\peek_meaning_remove:NTF	2996, 3348
\peek_remove_spaces:n	2137
\phantomsection	28
\phantomsection	261

prg commands:

`\prg_do_nothing:` 265
`\prg_new_protected_conditional:Npnn` ... 199
`\prg_replicate:nn` 208, 3712
`\prg_return_false:` 203
`\prg_return_true:` 202
`\printkeyans` 13, 96, 3539
prop commands:
`\prop_count:N` 1548, 1809, 1845, 1908, 1989
`\prop_gput:Nnn` 1546
`\prop_if_exist:NTF` 1544, 3505
`\prop_item:Nn` 3508
`\prop_new:N` 1545
`\ProvidesExplPackage` 4

R

`\raggedcolumns` 2450, 2590
`\ref` 59, 61
`ref` 464, 484
`\refstepcounter` 3059, 3407
regex commands:
`\regex_match:nnTF` 201, 524, 526, 538, 540, 2351, 2364, 2909, 2922
`\regex_replace_once:nnN` 400
`\renewcommand` 412, 433, 443
`\RenewDocumentCommand` ... 2029, 2099, 2133, 2159, 2175
`\RequirePackage` 17
`resume` 1322
`resume*` 1322
`rightmargin` 611
`\Roman` 29, 33, 34
`\Roman` 308
`\roman` 29, 33, 34
`\roman` 309, 482, 3527

S

`save-ans` 1322
scan commands:
`\scan_stop:` 80, 2716, 2866, 3289, 3496, 3499
seq commands:
`\seq_clear:N` 3602
`\seq_const_from_clist:Nn` 3590
`\seq_count:N` 2676, 3606
`\seq_gclear:N` 2027, 2028
`\seq_gput_right:Nn` 1556, 2040, 2041
`\seq_if_empty:NTF` 2046, 3554, 3620
`\seq_if_exist:NTF` 1554, 3552
`\seq_item:Nn` 2734
`\seq_map_function:NN` 3611
`\seq_map_inline:Nn` .. 3560, 3565, 3599, 3621, 3622
`\seq_map_pairwise_function:NNN` 2048
`\seq_new:N` 92, 93, 110, 140, 141, 1555
`\seq_pop_left:NN` 3610
`\seq_put_right:Nn` 2637, 3618, 3635
`\seq_set_from_clist:Nn` 3603
`\seq_set_map_e:NNn` 3612
`\seq_show:N` 3556
`\setcounter` .. 535, 539, 541, 2237, 2239, 2284, 2286, 2681
`\setenumext` .. 5–8, 97, 3515, 3520, 3525, 3530, 3535, 3595
`\setlength` 1600, 1608
`show-ans` 1470, 1499
`show-length` 687
skip commands:
`\skip_add:Nn` . 826, 832, 838, 848, 852, 876, 880, 977, 983, 989, 999, 1003, 1025, 1078, 1082, 2702

`\skip_eval:n` 1599, 1607
`\skip_gset:Nn` 1098, 1102, 1106
`\skip_gzero_new:N` 1093, 1094
`\skip_horizontal:N` 659, 671, 683, 3092, 3106, 3426, 3440
`\skip_horizontal:n` ... 645, 1628, 1636, 2152, 2154, 3090, 3424
`\skip_if_eq:nnTF` . 824, 830, 836, 900, 934, 975, 981, 987, 1018, 1023, 1044, 1095, 1117, 1230, 1244, 1258, 1269, 1280, 1291, 1302, 1313
`\skip_new:N` 54, 55, 59, 60, 61, 62, 63, 114, 170
`\skip_set:Nn` . 809, 813, 862, 866, 903, 907, 911, 918, 922, 926, 937, 942, 946, 952, 957, 962, 1020, 1021, 1022, 1029, 1033, 1037, 1046, 1051, 1055, 1058, 1062, 1066, 1097, 1101, 1119, 1123, 1127, 1133, 1137, 1141, 2696, 2710
`\skip_set_eq:NN` 2232, 2278, 2279, 3109, 3110, 3443, 3444
`\skip_use:N` 811, 815, 850, 854, 858, 878, 882, 901, 920, 929, 935, 940, 944, 955, 959, 960, 965, 1001, 1005, 1031, 1231, 1235, 1238, 1245, 1249, 1252, 2462
`\skip_zero:N` 2280, 2441, 2584, 2707, 2708
`\skip_zero_new:N` 1013, 1014, 1015, 1092, 1114, 1115, 1116
`\c_zero_skip` . 824, 830, 836, 901, 935, 975, 981, 987, 1018, 1023, 1044, 1095, 1117, 1231, 1245, 1258, 1269, 1280, 1291, 1302, 1313
`\small` 3517, 3522, 3527, 3532, 3537
`\star` 2011
`start` 547
`\stepcounter` 2033, 2644
`store-ref` 1470
str commands:

`\c_backslash_str` 3721, 3730, 3731, 3735, 3736, 3740, 3741, 3772, 3773, 3777, 3782, 3783
`\c_colon_str` 1808, 1907, 3496
`\str_count:n` 208, 3712
`\str_if_eq:nnTF` 2242, 2290
`\str_if_eq_p:nn` 2235, 2283
`\str_if_in:nnTF` 3492
`\str_new:N` 109, 165
`\str_set:Nn` ... 388, 389, 390, 1481, 1482, 1504, 1505
`\string` 254
`\strutbox` 905, 909, 913, 924, 928, 939, 948, 954, 964, 977, 983, 989, 1020, 1021, 1022, 1025, 1035, 1039, 1048, 1055, 1060, 1068, 1097, 1098, 1101, 1108, 1121, 1129, 1135, 1143, 2712

T

TeX and \TeX 2_ε commands:

`\@auxout` 270
`\protected@write` 270
text commands:
`\text_expand:n` 3488
`\textasteriskcentered` 1478, 1492
`\thepage` 276
tl commands:
`\c_space_tl` 1865, 1934, 1979, 2002, 3381, 3679, 3694
`\tl_clear:N` 361, 367, 1682, 1855, 1924, 3378
`\tl_clear_new:N` 318
`\tl_const:Nn` 142, 302
`\tl_gclear:N` 1966, 2170, 2508, 2851, 3093, 3161, 3275, 3427
`\tl_gput_right:Nn` 303
`\tl_greplace_all:Nnn` 324

`\tl_gset:Nn` 1954, 2470, 2955, 3026
`\tl_gset_eq:NN` 320, 2079, 3086, 3420
`\tl_if_blank:nTF` 3084, 3418
`\tl_if_empty:nTF` . 418, 452, 458, 1563, 1594, 1708, 1962, 2149, 3633
`\tl_if_novalue:nTF` . . 1683, 1694, 1863, 1932, 1978, 2001, 2031, 2056, 2075, 2080, 2110, 2674, 2896, 3315, 3379, 3597
`\tl_map_inline:Nn` 321, 398
`\tl_new:N` 32, 39, 41, 42, 75, 76, 77, 83, 84, 85, 87, 88, 89, 90, 96, 97, 107, 108, 119, 131, 132, 133, 136, 144, 145, 148, 149, 164, 167
`\tl_put_left:Nn` 1571, 1604, 1691, 1972, 1995, 3389, 3392
`\tl_put_right:Nn` 319, 410, 431, 441, 1523, 1530, 1575, 1610, 1693, 1699, 1707, 1710, 1720, 1725, 1728, 1734, 1760, 1770, 1784, 1800, 1806, 1811, 1858, 1861, 1865, 1892, 1897, 1902, 1905, 1914, 1927, 1930, 1934, 1944, 1979, 2002, 2356, 2369, 2914, 2927, 3513, 3518, 3523, 3528, 3533
`\tl_remove_all:Nn` 3632
`\tl_remove_once:Nn` 1748, 1877
`\tl_replace_all:Nnn` 323
`\tl_reverse:N` 1747, 1749, 1876, 1878
`\tl_set:Nn` 150, 288, 362, 366, 371, 372, 406, 425, 642, 656, 668, 680, 1340, 1477, 1491, 1839, 1983, 2077, 3381, 3630
`\tl_set_eq:NN` 329, 407, 409, 428, 430, 438, 440, 1746, 1875, 1888, 2122, 2126, 2655, 2657
`\tl_to_str:n` 3488
`\tl_trim_spaces:n` 319, 3618, 3630, 3636

`\tl_use:N` . 325, 328, 420, 454, 460, 713, 717, 721, 725, 729, 733, 737, 741, 745, 749, 753, 757, 761, 765, 769, 773, 1633, 1753, 1761, 1772, 1786, 1791, 1803, 2064, 2070, 2095, 2113, 2117, 2125, 2153, 2161, 2162, 2169, 2177, 2178, 2184, 2311, 2516, 2660, 2845, 3096, 3107, 3111, 3269, 3430, 3441, 3445, 3542, 3543, 3544, 3545, 3546, 3614

token commands:

`\token_to_str:N` 272
`\topsep` 1600, 1608
`topsep` 567
`\typeout` 240, 243, 253, 254

U

`\u` 401
use commands:
`\use:N` 209, 2166, 2313
`\use:n` 3494
`\use_none:nn` 264
`\usecounter` 2233, 2281

V

`\value` 2490, 2495, 2945, 2950
`\vspace` 892, 1235, 1238, 1249, 1252, 1262, 1264, 1273, 1275, 1284, 1286, 1295, 1297, 1306, 1308, 1317, 1319, 1599, 1607, 2671, 2682, 3147, 3481

W

`widest` 547
`wrap-ans` 1470
`wrap-label` 332
`wrap-label*` 332