

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 MiK \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)



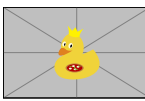
(B)



(C)



(D)



(E)

5. Question with image on left side:

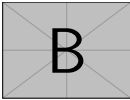
(A) value

(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.

<code>enumext.sty</code>	<code>>> TDS:tex/latex/enumext/</code>
<code>enumext.pdf</code>	<code>>> TDS:doc/latex/enumext/</code>
<code>README.md</code>	<code>>> TDS:doc/latex/enumext/</code>
<code>enumext.dtx</code>	<code>>> TDS:source/latex/enumext/</code>

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.

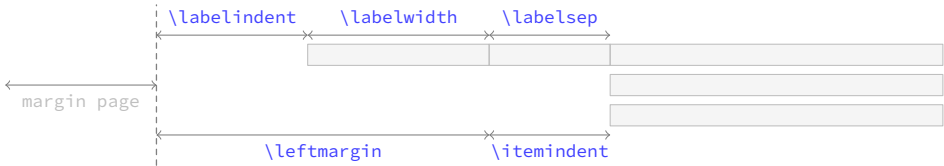


Figure 1: Representation of horizontal lengths in `enumitem`.

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.

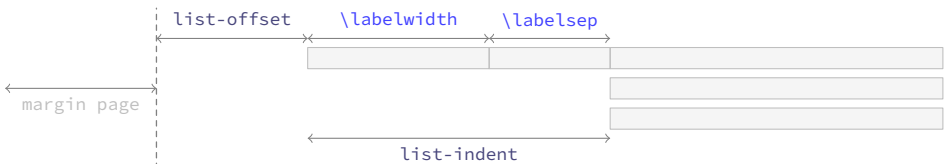


Figure 2: Representation of horizontal lengths concept in `enumext`.

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 $[\langle key = val \rangle]$ 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.

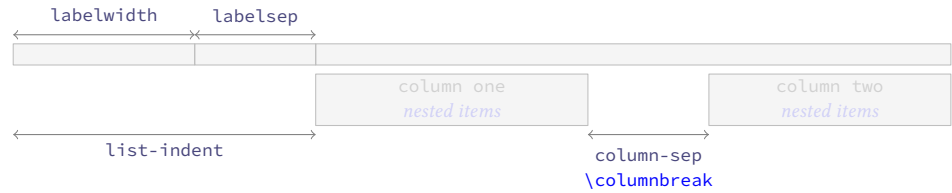


Figure 4: Representation of the two column output for a nested level in `enumext` environment.

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 `\mini-right` (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

<code>enumext</code>	<code>\begin{enumext} [⟨keyval list⟩]</code>	<code>\begin{enumext*} [⟨keyval list⟩]</code>
<code>enumext*</code>	<code>\item ⟨item content⟩</code>	<code>\item ⟨item content⟩</code>
	<code>\item [⟨custom⟩] ⟨item content⟩</code>	<code>\item [⟨custom⟩] ⟨item content⟩</code>
	<code>\item* [⟨symbol⟩] [⟨offset⟩] ⟨item content⟩</code>	<code>\item* [⟨symbol⟩] [⟨offset⟩] ⟨item content⟩</code>
	<code>\end{enumext}</code>	<code>\end{enumext*}</code>

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 $\langle key \rangle$ 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 $\langle keys \rangle$ 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 = { $\langle \backslash alph* | \backslash Alph* | \backslash arabic* | \backslash roman* | \backslash Roman* \rangle$ }` default: *by levels*

Sets the $\langle label \rangle$ 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 $\langle label \rangle$ 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 $\langle label \rangle$ as an argument, for example `\emph{\langle \backslash alph* \rangle}` will return an error. For full customization of how $\langle label \rangle$ is displayed use the `font` or `wrap-label` keys.

`ref = { $\langle code \{ \backslash alph* | \backslash Alph* | \backslash arabic* | \backslash roman* | \backslash Roman* \} more code \rangle$ }` 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{\langle \backslash alph* \rangle}` 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 = { $\langle rigid length \rangle$ }` default: `0.3333em`

Sets the *horizontal space* between the box containing the current $\langle label \rangle$ 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 = { $\langle rigid length \rangle$ }` default: *by label*

Sets the *width* of the box containing the current $\langle label \rangle$ 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 = { $\langle integer | string \rangle$ }` default: *empty*

Sets the `labelwidth` key pass the $\langle integer \rangle$ or converting the $\langle string \rangle$ 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 = { $\langle font commands \rangle$ }` default: *empty*

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

`align = { $\langle left | right | center \rangle$ }` default: *left*

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

`wrap-label = { $\langle code \{ \#1 \} more code \rangle$ }` default: *empty*

Wraps the current $\langle label \rangle$ defined by `label` key referenced by $\{ \#1 \}$. The $\{ \langle code \rangle \}$ 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* = { $\langle code \{ \#1 \} more code \rangle$ }` default: *empty*

The same as the `wrap-label` key but also applies on `\item[$\langle custom \rangle$]`.

3.2 Keys for spaces

`show-length = { $\langle true | false \rangle$ }` default: *false*

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

3.2.1 Vertical spaces

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

default: *by levels*

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

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

default: *by levels*

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

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

default: *by levels*

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

- The value of this parameter also affects the *inner levels* and the *keyans* environment. Caution should be taken with “blank lines” or `\par` command “before” each environment or nested level when formatting the source code of document. T_EX will enter *vertical mode* and apply this value to the “top” and “bottom” the environment or nested level.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

3.2.2 Horizontal spaces

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

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

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

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

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

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

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

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

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

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

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

3.3 Keys for add code

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

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

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

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

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

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

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

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

3.4 Keys for start and resume

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

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

3.5 Keys for multicol

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

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

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

3.6 Keys for minipage

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

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

3.6.1 The command `\miniright`

`\miniright` The `\miniright` command close the `minipage` environment on the “left side” and opens the `minipage`
`\miniright*` environment on the “right side” by starting it with the `\centering` command. It must be placed “after” the last `\item` of the current environment and “before” starting the material to be placed on the “right side”. The starred version ‘*’ inhibits the use of `\centering` command i.e. the usual L^AT_EX justification is maintained in the `minipage` on the “right side”.

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

4 The storage system

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

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

4.1 Keys for storage

- save-ans = { <store name> }

default: not set

Sets the “name” of the <sequence> and <prop list> in which the contents will be “stored” by \anskey in enumext environment, \item* in keyans environment and \anspic* in keyanspic environment. If the <sequence> or <prop list> does not exist, it will be created globally.
- wrap-ans = { <code {#1} more code> }

default: \fbox

Wraps the current <argument> passed \anskey command to referenced by {#1}. The {<code>} must be passed between braces. This <key> only affects the current <argument> passed to \anskey and NOT the “stored content” in the <store name> set by save-ans key. If this key is passed using the \setenumext command it is necessary to use double ‘{##1}’.
- mark-ans = { <symbol> }

default: \textasteriskcentered

Sets the symbol to be displayed in the left margin of the “stored content” in <store name> set by save-ans key when using show-ans key.
- mark-pos = { <left | right> }

default: left

Sets the aligned of the symbol defined by mark-ans key. The “symbol” is aligned in a box with the same dimensions of the label box defined by labelwidth key on the current level and separated by the value of the labelsep key.
- show-ans = { <true | false> }

default: false

Displays the current <argument> passed to \anskey in enumext environment, the current <label> for \item* in keyans environment and the current <label> for \anspic* in keyanspic environment at the place where it is executed. If the optional argument is present in \item* or \anspic* it will be shown in square brackets.
- show-pos = { <true | false> }

default: false

Displays the position occupied by the “stored content” by \anskey in enumext environment, \item* in keyans environment and \anspic* in keyanspic environment in <store name> set by save-ans key. This position is used by the \getkeyans command and by the \ref command if the store-ref key is active.

4.2 Keys for internal label and ref

- store-ref = { <true | false> }

default: false

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

default: \textasteriskcentered

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

4.3 Keys for check answers

- check-ans = { <true | false> }

default: false

Enables the “checking answer” mechanism. This key works under the logic that each question will contain “only one answer”, it is intended to be used in conjunction with no-store key.
- no-store <value forbidden>

default: not used

This is a “meta-key” that does not receive an argument. This key is used in conjunction with check-ans and is designed to be used with nested levels of enumext in which the \anskey command will not be used.

4.4 The command \anskey

\anskey \anskey{<content>}

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

Example

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

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

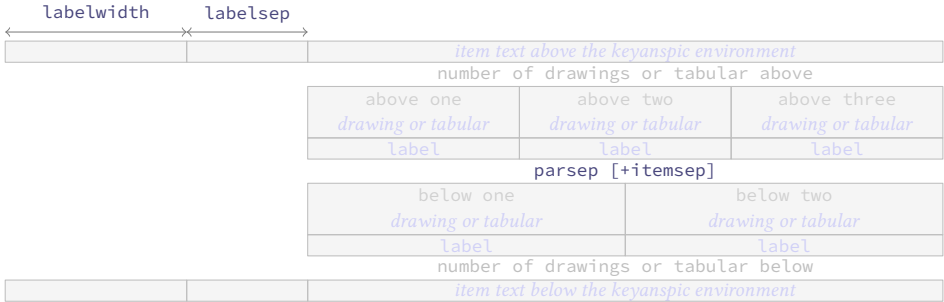


Figure 6: Representation of the `keyanspic` environment with optional argument `[3,2]` in `enumext`.

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

4.6.1 The command \anspic

\anspic

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

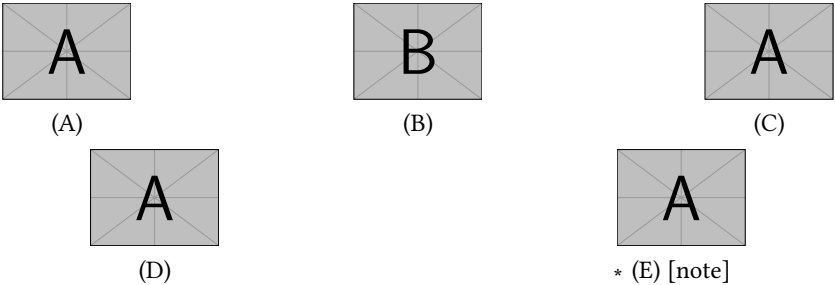
The `\anspic` command take three arguments, the *starred version* “*” store the current `\label` next to the `\content` (if it is present) in `\store name` set by `save-ans` key.

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

Example

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

1. Question with images.



4.7 Printing stored content

4.7.1 The command \getkeyans

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

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

The “content” can only be accessed “after” it is stored, if the `<store name>` does not exist the command will return an error. The form taken by the argument `<store name> : <position>` is the same as that used to generate the internal “label and ref” system when `store-ref` key are active, so to refer to a stored “content”. For example `\getkeyans{test:4}` will return the “stored content” at position 4 of the environment in which the key `save-ans=test` was set.

4.7.2 The command \printkeyans

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

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

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

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

Example

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

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

  \item Related to Linux

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

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

```
\printkeyans{sample}
```

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

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

2. True False

(a) \LaTeXe is cool?

[2] Very True!

3. Related to Linux

(a) You use linux?

[3] Yes

(b) Rate the following package and class

i. `xsim`

[4] very good

ii. `exsheets`

[5] obsolete

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

1. (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

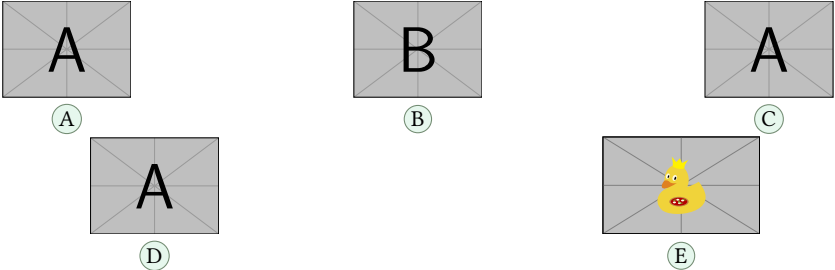
 value

D

 value

E

 value
4. Question with image and label below:



5. Question with image on left side:
- A

 value

B

 value

C

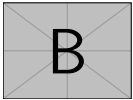
 value

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) ~~ETX~~ze 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	T
\ref 4	\topsep 7

10 Implementation

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

- The documentation presented here is far from professional, it contains a lot of obvious information that to the eye of a T_EXpert 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 (L^AT_EX3 DocStrip convention) for l3doc class.

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

10.3 Declaration of the package

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

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

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 \__enumext_level_int
21 \int_new:N \__enumext_level_h_int
22 \int_new:N \__enumext_keyans_level_int
23 \int_new:N \__enumext_keyans_level_h_int
24 \int_new:N \__enumext_keyans_pic_level_int
25 \bool_new:N \__enumext_starred_bool
26 \bool_new:N \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
\l__enumext_keyans_tmppb_tl
\l__enumext_keyans_tmpa_dim
```

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
89 \tl_new:N \l__enumext_keyans_tmppb_tl
90 \dim_new:N \l__enumext_keyans_tmpa_dim
```

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

```
\l__enumext_setkey_tmpa_tl
\l__enumext_setkey_tmppb_tl
\l__enumext_setkey_tmpa_int
\l__enumext_setkey_tmpa_seq
\l__enumext_setkey_tmppb_seq
```

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

```
91 \tl_new:N \l__enumext_setkey_tmpa_tl
92 \tl_new:N \l__enumext_setkey_tmppb_tl
93 \int_new:N \l__enumext_setkey_tmpa_int
94 \seq_new:N \l__enumext_setkey_tmpa_seq
95 \seq_new:N \l__enumext_setkey_tmppb_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*`.

```
96 \cs_set_protected:Npn \l__enumext_tmp:n #1
97 {
98   \tl_new:c { \l__enumext_store_opt_#1_tl }
99   \tl_new:c { \l__enumext_print_keyans_#1_tl }
100   \bool_new:c { \l__enumext_store_columns_#1_bool }
101   \int_new:c { \l__enumext_store_columns_#1_int }
102   \bool_new:c { \l__enumext_store_columns_sep_#1_bool }
103   \dim_new:c { \l__enumext_store_columns_sep_#1_dim }
104   \bool_new:c { \l__enumext_store_upper_level_#1_bool }
105 }
106 \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).

```
107 \bool_new:N \l__enumext_show_answer_bool
108 \bool_new:N \l__enumext_show_position_bool
109 \tl_new:N \l__enumext_mark_ref_sym_tl
110 \tl_new:N \l__enumext_mark_answer_sym_tl
111 \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).

```
112 \seq_new:N \l__enumext_keyans_pic_body_seq
113 \dim_new:N \l__enumext_keyans_pic_width_dim
114 \int_new:N \l__enumext_keyans_pic_above_int
115 \int_new:N \l__enumext_keyans_pic_below_int
116 \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.

```
117 \bool_new:N \l__enumext_store_ans_bool
118 \bool_new:N \l__enumext_check_ans_bool
119 \bool_new:N \g__enumext_check_ans_show_bool
120 \bool_new:N \g__enumext_check_ans_show_h_bool
121 \tl_new:N \g__enumext_check_ans_item_tl
122 \int_new:N \l__enumext_compare_items_ans_int
123 \int_new:N \g__enumext_count_item_with_ans_int
124 \int_new:N \g__enumext_count_item_all_int
125 \cs_set_protected:Npn \l__enumext_tmp:n #1
126 {
127   \int_new:c { \g__enumext_count_level_#1_int }
128   \int_new:c { \g__enumext_count_item_#1_int }
129 }
130 \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`.

```
131 \bool_new:N \l__enumext_hyperref_bool
132 \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.

```

133 \tl_new:N \l__enumext_newlabel_arg_one_tl
134 \tl_new:N \l__enumext_newlabel_arg_two_tl
135 \tl_new:N \l__enumext_store_write_aux_file_tl
136 \cs_set_protected:Npn \__enumext_tmp:n #1
137 {
138   \tl_new:c { l__enumext_label_copy_#1_tl }
139 }
140 \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`.

```

141 \int_new:N \g__enumext_footnote_int
142 \seq_new:N \g__enumext_footnote_arg_seq
143 \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).

```

144 \tl_const:Nn \c__enumext_counter_style_tl
145   { { arabic } { roman } { Roman } { alph } { Alph } }
146 \tl_new:N \l__enumext_ref_key_arg_tl
147 \tl_new:N \l__enumext_ref_aux_tl
148 \cs_set_protected:Npn \__enumext_tmp:n #1
149 {
150   \tl_new:c { l__enumext_counter_style_for_ref_#1_tl }
151   \tl_new:c { l__enumext_the_counter_#1_tl }
152   \tl_set:ce { l__enumext_the_counter_#1_tl } { \exp_not:c { theenumX#1 } }
153 }
154 \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.

```

155 \cs_set_protected:Npn \__enumext_tmp:n #1
156 {
157   \bool_new:c { l__enumext_item_starred_#1_bool }
158   \int_new:c { l__enumext_item_column_pos_#1_int }
159   \int_new:c { g__enumext_item_count_all_#1_int }
160   \int_new:c { l__enumext_joined_item_#1_int }
161   \int_new:c { l__enumext_joined_item_aux_#1_int }
162   \int_new:c { l__enumext_tmpa_#1_int }
163   \box_new:c { l__enumext_item_text_#1_box }
164   \dim_new:c { l__enumext_joined_width_#1_dim }
165   \dim_new:c { l__enumext_item_width_#1_dim }
166   \tl_new:c { g__enumext_item_symbol_aux_#1_tl }
167   \str_new:c { l__enumext_align_label_#1_str }
168   \bool_new:c { g__enumext_minipage_active_#1_bool }
169   \tl_new:c { g__enumext_miniright_code_#1_tl }
170   \bool_new:c { g__enumext_minipage_center_#1_bool }
171   \dim_new:c { g__enumext_minipage_right_#1_dim }
172   \skip_new:c { g__enumext_minipage_right_#1_skip }
173 }
174 \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`.

```

175 \clist_const:Nn \c__enumext_all_envs_clist
176 {
177   {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv},
178   {keyans}{v}, {enumext*}{vii}, {keyans*}{viii}
179 }

```

(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.

```
180 \cs_new_protected:Npn \__enumext_at_begin_document:n #1
181 {
182   \hook_gput_code:nnn {begindocument} {enumext} { #1 }
183 }
```

(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.

```
184 \cs_new_protected:Npn \__enumext_after_env:nn #1 #2
185 {
186   \hook_gput_code:nnn {env/#1/after} {enumext} {#2}
187 }
```

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

`__enumext_level:`

Function for check current level in `enumext`.

```
188 \cs_new:Nn \__enumext_level:
189 {
190   \int_to_roman:n { \__enumext_level_int }
191 }
```

(End of definition for `__enumext_level:`.)

`__enumext_level_set:n`

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

`__enumext_level_end:n`

```
192 \cs_new:Npn \__enumext_level_set:n #1
193 {
194   \cs_set_eq:cN { \__enumext_level_#1: } \__enumext_level:
195   \cs_set:Nn \__enumext_level: { #1 }
196 }
197 \cs_new:Npn \__enumext_level_end:n #1
198 {
199   \cs_set_eq:Nc \__enumext_level: { __enumext_level_#1: }
200 }
```

(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`

```
201 \prg_new_protected_conditional:Npnn \__enumext_if_is_int:n #1 { T, F, TF }
202 {
203   \regex_match:nnTF { ^[\+|-]?[\d]+$ } {#1} % $
204   { \prg_return_true: }
205   { \prg_return_false: }
206 }
```

(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.

```
207 \cs_new:Npn \__enumext_show_length:nnn #1 #2 #3
208 {
209   * ~ #2
210   \prg_replicate:nn { 14 - \str_count:n {#2} } { ~ }
211   = ~ \use:c { #1_use:c } { \__enumext_#2_#3_#1 } \\
212 }
```

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

`__enumext_zero_count_level:`

Internal function used by `check-ans` key.

```
213 \cs_set_protected:Nn \__enumext_zero_count_level:
214 {
215   \cs_set_protected:Npn \__enumext_tmp:n ##1
216   {
217     \int_gzero:c { g__enumext_count_level_##1_int }
218   }
219   \clist_map_inline:nn { i, ii, iii, iv, vii } { \__enumext_tmp:n {##1} }
220 }
```

(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.

```
221 \__enumext_at_begin_document:n
222 {
223   \cs_new_eq:NN \__enumext_start_list:nn \list
224   \cs_new_eq:NN \__enumext_stop_list: \endlist
225   \cs_new_eq:NN \__enumext_item_std:w \item
226 }
```

(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.

```
227 \__enumext_at_begin_document:n
228 {
229   \cs_new_eq:NN \__enumext_minipage:w \minipage
230   \cs_new_eq:NN \__enumext_endminipage: \endminipage
231 }
```

(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.

```
232 \hook_gput_code:nnn { begindocument } { enumext } { \__enumext_after_hyperref: }
233 \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”.

```
234 \cs_new_protected:Nn \__enumext_after_hyperref:
235 {
236   \IfPackageLoadedTF { hyperref }
237   {
238     \msg_info:nnn { enumext } { package-load } { hyperref }
239     \bool_set_true:N \l__enumext_hyperref_bool
240     \IfHyperBoolean{hyperfootnotes}
241     {
242       \typeout{hyperfootnotes=true}
243       \bool_set_true:N \l__enumext_footnotes_key_bool
244     }
245     { \typeout{hyperfootnotes=false} }
246   }
247   { }
```

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`.

```
248 \bool_if:NT \l__enumext_footnotes_key_bool
249 {
250   \IfPackageLoadedTF { footnotehyper }
251   {
```

```

252         \msg_info:nnn { enumext } { package-load } { footnotehyper }
253     }
254     {
255         \typeout{No ~ footnotehyper ~ load}
256         \typeout{Load ~ and ~ use ~ \string\makesavenoteenv{enumext*}}
257         \bool_set_false:N \l__enumext_footnotes_key_bool
258     }
259 }

```

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.

```

260     \bool_if:NTF \l__enumext_hyperref_bool
261     {
262         \cs_new_eq:NN \__enumext_hypertarget:nn \hypertarget
263         \cs_new_eq:NN \__enumext_phantomsection: \phantomsection
264     }
265     {
266         \cs_new_eq:NN \__enumext_hypertarget:nn \use_none:nn
267         \cs_new_eq:NN \__enumext_phantomsection: \prg_do_nothing:
268     }
269 }

```

(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.

```

270 \cs_new_protected:Npn \__enumext_newlabel:nn #1 #2
271 {
272     \protected@write \@auxout { }
273     {
274         \token_to_str:N \newlabel {#1}
275         {
276             {#2}
277             \bool_if:NT \l__enumext_hyperref_bool
278             { { \thepage } {#2} {#1} }
279             { }
280         }
281     }
282     \__enumext_hypertarget:nn {#1} { }
283     \__enumext_phantomsection:
284 }

```

(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.

```

285 \cs_new_protected:Npn \__enumext_define_counters:Nn #1 #2
286 {
287     \cs_if_exist:cTF { c@ #2 }
288     { \msg_fatal:nnn { enumext } { counters } { #2 } }
289     {
290         \tl_set:Nn #1 { #2 }
291         \newcounter { #2 }
292     }
293 }

```

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

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

```
312 \cs_new_protected:Npn \__enumext_label_width_by_box:Nn #1 #2
313 {
314   \hbox_set:Nn \l__enumext_label_width_by_box {#2}
315   \dim_set:Nn #1 { \box_wd:N \l__enumext_label_width_by_box }
316 }
317 \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.

```
318 \cs_new_protected:Npn \__enumext_label_style:Nnn #1 #2 #3
319 {
320   \tl_clear_new:N #1
321   \tl_put_right:Ne #1 { \tl_trim_spaces:n {#3} }
322   \tl_gset_eq:NN \g__enumext_widest_label_tl #1
323   \tl_map_inline:Nn \g__enumext_counter_styles_tl
324   {
325     \tl_replace_all:Nne #1 { ##1* } { \exp_not:N ##1 {#2} }
326     \tl_greplace_all:Nne \g__enumext_widest_label_tl { ##1* }
327     { \tl_use:c { c__enumext_widest_ \cs_to_str:N ##1 _tl } }
328   }
329   \__enumext_label_width_by_box:Nn \l__enumext_current_widest_dim
330   { \tl_use:N \g__enumext_widest_label_tl }
331   \tl_set_eq:cN { the #2 } #1
332 }
333 \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.

```

labelsep
labelwidth
wrap-label
wrap-label*
334 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
335 {
336   \keys_define:nn { enumext / #1 }
337   {
338     font .tl_set:c = { l__enumext_label_font_style_#2_tl },
339     font .value_required:n = true,
340     labelsep .dim_set:c = { l__enumext_labelsep_#2_dim },
341     labelsep .initial:n = {0.3333em},
342     labelsep .value_required:n = true,
343     labelwidth .dim_set:c = { l__enumext_labelwidth_#2_dim },
344     labelwidth .value_required:n = true,
345     wrap-label .cs_set_protected:cp = { __enumext_wrapper_label_#2:n } ##1,
346     wrap-label .initial:n = {##1},
347     wrap-label .value_required:n = true,
348     wrap-label* .code:n = {
349       \bool_set_true:c { l__enumext_wrap_label_opt_#2_bool }
350       \keys_set:nn { enumext / #1 } { wrap-label = {##1} }
351     },
352     wrap-label* .value_required:n = true,
353   }
354 }
355 \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.

```

356 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
357 {
358   \keys_define:nn { enumext / #1 }
359   {
360     align .choice:,
361     align / left .code:n =
362       {
363         \tl_clear:c { l__enumext_label_fill_left_#2_tl }
364         \tl_set:cn { l__enumext_label_fill_right_#2_tl } { \hfill }
365       },
366     align / right .code:n =
367       {
368         \tl_set:cn { l__enumext_label_fill_left_#2_tl } { \hfill }
369         \tl_clear:c { l__enumext_label_fill_right_#2_tl }
370       },
371     align / center .code:n =
372       {
373         \tl_set:cn { l__enumext_label_fill_left_#2_tl } { \hfill }
374         \tl_set:cn { l__enumext_label_fill_right_#2_tl } { \hfill }
375       },
376     align .initial:n = left,
377     align .value_required:n = true,
378   }
379 }
380 \clist_map_inline:nn
381 {
382   {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {keyans}{v}
383 }
384 { \__enumext_tmp:nn #1 }
```

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

```

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

```

392     align / center .code:n = \str_set:cn { \__enumext_align_label_#2_str } { c },
393     align .initial:n = left,
394     align .value_required:n = true,
395   }
396 }
397 \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.

```

398 \cs_new_protected:Nn \__enumext_regex_label_ref_key:
399 {
400   \tl_map_inline:Nn \c__enumext_counter_style_tl
401   {
402     \regex_replace_once:nnN { \c{##1}\* }
403     { \c{##1}\cB{\u{\__enumext_ref_aux_tl}\cE} } \__enumext_ref_key_arg_tl
404   }
405 }

```

(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`.

```

406 \cs_new_protected:Npn \__enumext_set_label_ref:n #1
407 {
408   \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
409   \tl_set_eq:Nc \l__enumext_ref_aux_tl { \__enumext_counter_ \__enumext_level: _tl }
410   \__enumext_regex_label_ref_key:
411   \tl_set_eq:Nc \l__enumext_ref_aux_tl { \__enumext_the_counter_ \__enumext_level: _tl }
412   \tl_put_right:ce { \__enumext_counter_style_for_ref_ \__enumext_level: _tl }
413   {
414     \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_aux_tl }
415     { \exp_not:V \l__enumext_ref_key_arg_tl }
416   }
417 }

```

(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`.

```

418 \cs_new_protected:Nn \__enumext_use_key_ref:
419 {
420   \tl_if_empty:cF { \__enumext_counter_style_for_ref_ \__enumext_level: _tl }
421   {
422     \tl_use:c { \__enumext_counter_style_for_ref_ \__enumext_level: _tl }
423   }
424 }

```

(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`.

```

425 \cs_new_protected:Npn \__enumext_set_label_ref_h:n #1
426 {

```

```

427 \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
428 \int_compare:nNnTF { \l__enumext_level_h_int } = { 1 }
429 {
430   \tl_set_eq:NN \l__enumext_ref_aux_tl \l__enumext_counter_vii_tl
431   \__enumext_regex_label_ref_key:
432   \tl_set_eq:NN \l__enumext_ref_aux_tl \l__enumext_the_counter_vii_tl
433   \tl_put_right:Ne \l__enumext_counter_style_for_ref_vii_tl
434   {
435     \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_aux_tl }
436     { \exp_not:V \l__enumext_ref_key_arg_tl }
437   }
438 }
439 {
440   \tl_set_eq:NN \l__enumext_ref_aux_tl \l__enumext_counter_viii_tl
441   \__enumext_regex_label_ref_key:
442   \tl_set_eq:NN \l__enumext_ref_aux_tl \l__enumext_the_counter_viii_tl
443   \tl_put_right:Ne \l__enumext_counter_style_for_ref_viii_tl
444   {
445     \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_aux_tl }
446     { \exp_not:V \l__enumext_ref_key_arg_tl }
447   }
448 }
449 }

```

(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*`.

```

450 \cs_new_protected:Nn \__enumext_use_key_ref_h:
451 {
452   \int_compare:nNnTF { \l__enumext_level_h_int } = { 1 }
453   {
454     \tl_if_empty:NF \l__enumext_counter_style_for_ref_vii_tl
455     {
456       \tl_use:N \l__enumext_counter_style_for_ref_vii_tl
457     }
458   }
459   {
460     \tl_if_empty:NF \l__enumext_counter_style_for_ref_viii_tl
461     {
462       \tl_use:N \l__enumext_counter_style_for_ref_viii_tl
463     }
464   }
465 }

```

(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.

```

\__enumext_label_i_tl 466 \cs_set_protected:Npn \__enumext_tmp:nnn #1 #2 #3
\__enumext_label_ii_tl 467 {
\__enumext_label_iii_tl 468   \keys_define:nn { enumext / #1 }
\__enumext_label_iv_tl 469   {
470     label .code:n = {
471       \__enumext_label_style:cvn { l__enumext_label_#2_tl }
472       { l__enumext_counter_#2_tl } {##1}
473       \dim_set_eq:cN { l__enumext_labelwidth_#2_dim }
474       \l__enumext_current_widest_dim
475     },
476     label .initial:n = #3,
477     label .value_required:n = true,
478     ref .code:n = \__enumext_set_label_ref:n {##1},
479     ref .value_required:n = true,
480   }
481 }
482 \__enumext_tmp:nnn { level-1 } { i } { \arabic*. }
483 \__enumext_tmp:nnn { level-2 } { ii } { (\alph*) }
484 \__enumext_tmp:nnn { level-3 } { iii } { \roman*. }
485 \__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
486 \cs_set_protected:Npn \__enumext_tmp:nnn #1 #2 #3
487 {
488   \keys_define:nn { enumext / #1 }
489   {
490     label .code:n = {
491       \__enumext_label_style:cvn { l__enumext_label_#2_tl }
492       { l__enumext_counter_#2_tl } {##1}
493       \dim_set_eq:cN { l__enumext_labelwidth_#2_dim }
494       \l__enumext_current_widest_dim
495     },
496     label .initial:n = #3,
497     label .value_required:n = true,
498     ref .code:n = \__enumext_set_label_ref_h:n {##1},
499     ref .value_required:n = true,
500   }
501 }
502 \__enumext_tmp:nnn { enumext* } { vii } { \arabic*.}
503 \__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.

```

504 \keys_define:nn { enumext / keyans }
505 {
506   label .code:n = {
507     \__enumext_label_style:cvn { l__enumext_label_v_tl }
508     { l__enumext_counter_v_tl } {##1}
509     \dim_set_eq:cN { l__enumext_labelwidth_v_dim }
510     \l__enumext_current_widest_dim
511     \__enumext_label_style:cvn { l__enumext_label_vi_tl }
512     { l__enumext_counter_vi_tl } {##1}
513     \dim_set_eq:cN { l__enumext_labelwidth_v_dim }
514     \l__enumext_current_widest_dim
515   },
516   label .initial:n = (\Alph*),
517   label .value_required:n = true,
518 }

```

(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.

```

519 \cs_new_protected:Npn \__enumext_start_from:NNn #1 #2 #3
520 {
521   \__enumext_if_is_int:nTF { #3 }
522   {
523     \int_set:Nn #2 {##3}
524   }
525   {
526     \regex_match:nVT { \c{Alph} | \c{alph} } {##1}
527     { \int_set:Nn #2 { \int_from_alph:n {##3} } }
528     \regex_match:nVT { \c{Roman} | \c{roman} } {##1}
529     { \int_set:Nn #2 { \int_from_roman:n {##3} } }
530   }
531 }
532 \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.

```
533 \cs_new_protected:Npn \__enumext_widest_from:nNNn #1 #2 #3 #4
534 {
535   \__enumext_if_is_int:nTF {#4}
536   {
537     \setcounter{enumX#1} { #4 }
538   }
539   {
540     \regex_match:nVT { \c{Alph} | \c{alph} } {#2}
541     { \setcounter{enumX#1} { \int_from_alph:n {#4} } }
542     \regex_match:nVT { \c{Roman} | \c{roman} } {#2}
543     { \setcounter{enumX#1} { \int_from_roman:n {#4} } }
544   }
545   \__enumext_label_width_by_box:cv
546   { \l__enumext_labelwidth_#1_dim } { \l__enumext_label_#1_tl }
547 }
548 \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
```

```
549 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
550 {
551   \keys_define:nn { enumext / #1 }
552   {
553     start .code:n = {
554       \__enumext_start_from:ccn
555       { \l__enumext_label_#2_tl }
556       { \l__enumext_start_#2_int } {##1}
557     },
558     start .initial:n = 1,
559     widest .code:n = {
560       \__enumext_widest_from:nccn {#2}
561       { \l__enumext_label_#2_tl }
562       { \l__enumext_labelwidth_#2_dim } {##1}
563     },
564     widest .value_required:n = true,
565     start .value_required:n = true,
566   }
567 }
568 \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.

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

```

581     parsep      .value_required:n = true,
582     itemsep     .skip_set:c = { l__enumext_itemsep_#2_skip },
583     itemsep     .initial:n = {#6},
584     itemsep     .value_required:n = true,
585     noitemsep   .meta:n = { itemsep = 0pt, parsep = 0pt },
586     noitemsep   .value_forbidden:n = true,
587     nosepe     .meta:n = {
588                     itemsep = 0pt, parsep= 0pt,
589                     topsep = 0pt, partopsep = 0pt,
590                     },
591     nosepe     .value_forbidden:n = true,
592   }
593 }

```

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

```

594 \__enumext_tmp:nnnnnn { level-1 } { i } { 8.0pt plus 2.0pt minus 4.0pt }
595 { 2.0pt plus 1.0pt minus 1.0pt } { 4.0pt plus 2.0pt minus 1.0pt }
596 { 4.0pt plus 2.0pt minus 1.0pt }
597 \__enumext_tmp:nnnnnn { level-2 } { ii } { 4.0pt plus 2.0pt minus 1.0pt }
598 { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
599 { 2.0pt plus 1.0pt minus 1.0pt }
600 \__enumext_tmp:nnnnnn { level-3 } { iii } { 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 { level-4 } { iv } { 2.0pt plus 1.0pt minus 1.0pt }
603 { 1.0pt minus 1.0pt } { 0pt } { 2.0pt plus 1.0pt minus 1.0pt }
604 \__enumext_tmp:nnnnnn { keyans } { v } { 4.0pt plus 2.0pt minus 1.0pt }
605 { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
606 { 2.0pt plus 1.0pt minus 1.0pt }
607 \__enumext_tmp:nnnnnn { enumext* } { vii } { 8.0pt plus 2.0pt minus 4.0pt }
608 { 2.0pt plus 1.0pt minus 1.0pt } { 4.0pt plus 2.0pt minus 1.0pt }
609 { 4.0pt plus 2.0pt minus 1.0pt }
610 \__enumext_tmp:nnnnnn { keyans* } { viii } { 4.0pt plus 2.0pt minus 1.0pt }
611 { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
612 { 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.

```

613 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
614 {
615   \keys_define:nn { enumext / #1 }
616   {
617     itemindent .dim_set:c = { l__enumext_fake_item_indent_#2_dim },
618     itemindent .value_required:n = true,
619     rightmargin .dim_set:c = { l__enumext_rightmargin_#2_dim },
620     rightmargin .value_required:n = true,
621     listparindent .dim_set:c = { l__enumext_listparindent_#2_dim },
622     listparindent .value_required:n = true,
623     list-offset .dim_set:c = { l__enumext_listoffset_#2_dim },
624     list-offset .value_required:n = true,
625     list-indent .code:n =
626       \bool_set_true:c { l__enumext_leftmargin_tmp_#2_bool }
627       \dim_set:cn { l__enumext_leftmargin_tmp_#2_dim } {##1},
628     list-indent .value_required:n = true,
629   }
630 }
631 \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.

```

632 \cs_set_protected:Npn \__enumext_tmp:n #1
633 {
634   \keys_define:nn { enumext / #1 } { list-indent .initial:n = 0pt, }
635 }
636 \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.

```

637 \cs_set_protected:Nn \__enumext_fake_item:
638 {
639   \dim_compare:nNnT
640     { \dim_use:c { l__enumext_fake_item_indent_ \__enumext_level: _dim } }
641     >
642     { \c_zero_dim }
643   {
644     \tl_set:ce { l__enumext_fake_item_indent_ \__enumext_level: _tl }
645     {
646       \exp_not:N \mode_leave_vertical:
647       \exp_not:n { \skip_horizontal:n }
648       { \dim_use:c { l__enumext_fake_item_indent_ \__enumext_level: _dim } }
649       \ignorespaces
650     }
651   }
652 }
653 \cs_set_protected:Nn \__enumext_keyans_fake_item:
654 {
655   \dim_compare:nNnT
656     { \l__enumext_fake_item_indent_v_dim } > { \c_zero_dim }
657   {
658     \tl_set:Ne \l__enumext_fake_item_indent_v_tl
659     {
660       \exp_not:N \mode_leave_vertical:
661       \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_v_dim
662     }
663   }
664 }
665 \cs_set_protected:Nn \__enumext_fake_item_vii:
666 {
667   \dim_compare:nNnT
668     { \l__enumext_fake_item_indent_vii_dim } > { \c_zero_dim }
669   {
670     \tl_set:Ne \l__enumext_fake_item_indent_vii_tl
671     {
672       \exp_not:N \mode_leave_vertical:
673       \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_vii_dim
674     }
675   }
676 }
677 \cs_set_protected:Nn \__enumext_fake_item_viii:
678 {
679   \dim_compare:nNnT
680     { \l__enumext_fake_item_indent_viii_dim } > { \c_zero_dim }
681   {
682     \tl_set:Ne \l__enumext_fake_item_indent_viii_tl
683     {
684       \exp_not:N \mode_leave_vertical:
685       \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_viii_dim
686     }
687   }
688 }

```

(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.

```

689 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
690 {
691   \keys_define:nn { enumext / #1 }
692   {
693     show-length .bool_set:c = { l__enumext_show_length_#2_bool },

```

```

694         show-length .initial:n = false,
695     }
696 }
697 \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* 698 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
after    699 {
first    700     \keys_define:nn { enumext / #1 }
          {
            before .tl_set:c = { l__enumext_before_no_starred_key_#2_tl },
            before .value_required:n = true,
            before* .tl_set:c = { l__enumext_before_starred_key_#2_tl },
            before* .value_required:n = true,
            after .tl_set:c = { l__enumext_after_stop_list_#2_tl },
            after .value_required:n = true,
            first .tl_set:c = { l__enumext_after_list_args_#2_tl },
            first .value_required:n = true,
          }
        }
        712 \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }

```

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

10.16.1 Functions for before, after and first keys in enumext

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

```

713 \cs_new_protected:Nn \__enumext_before_args_exec:
714 {
715     \tl_use:c { l__enumext_before_starred_key_ \__enumext_level: _tl }
716 }

```

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

```

717 \cs_new_protected:Nn \__enumext_before_keys_exec:
718 {
719     \tl_use:c { l__enumext_before_no_starred_key_ \__enumext_level: _tl }
720 }

```

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

```

721 \cs_new_protected:Nn \__enumext_after_stop_list:
722 {
723     \tl_use:c { l__enumext_after_stop_list_ \__enumext_level: _tl }
724 }

```

The function `__enumext_after_args_exec:` executes the `{⟨code⟩}` set by the `first` key after the end of the second argument of the list defining the `enumext` environment, just before the first occurrence of `\item`.

```

725 \cs_new_protected:Nn \__enumext_after_args_exec:
726 {
727     \tl_use:c { l__enumext_after_list_args_ \__enumext_level: _tl }
728 }

```

(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 `{⟨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.

```

729 \cs_new_protected:Nn \__enumext_before_args_exec_v:
730 {
731     \tl_use:N \l__enumext_before_starred_key_v_tl
732 }

```

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⟩`.

```
733 \cs_new_protected:Nn \__enumext_before_keys_exec_v:
734 {
735   \tl_use:N \l__enumext_before_no_starred_key_v_tl
736 }
```

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

```
737 \cs_new_protected:Nn \__enumext_after_stop_list_v:
738 {
739   \tl_use:N \l__enumext_after_stop_list_v_tl
740 }
```

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`.

```
741 \cs_new_protected:Nn \__enumext_after_args_exec_v:
742 {
743   \tl_use:N \l__enumext_after_list_args_v_tl
744 }
```

(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.

```
745 \cs_new_protected:Nn \__enumext_before_args_exec_vii:
746 {
747   \tl_use:N \l__enumext_before_starred_key_vii_tl
748 }
749 \cs_new_protected:Nn \__enumext_before_args_exec_viii:
750 {
751   \tl_use:N \l__enumext_before_starred_key_viii_tl
752 }
```

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⟩`.

```
753 \cs_new_protected:Nn \__enumext_before_keys_exec_vii:
754 {
755   \tl_use:N \l__enumext_before_no_starred_key_vii_tl
756 }
757 \cs_new_protected:Nn \__enumext_before_keys_exec_viii:
758 {
759   \tl_use:N \l__enumext_before_no_starred_key_viii_tl
760 }
```

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

```
761 \cs_new_protected:Nn \__enumext_after_stop_list_vii:
762 {
763   \tl_use:N \l__enumext_after_stop_list_vii_tl
764 }
765 \cs_new_protected:Nn \__enumext_after_stop_list_viii:
766 {
767   \tl_use:N \l__enumext_after_stop_list_viii_tl
768 }
```

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`.

```
769 \cs_new_protected:Nn \__enumext_after_args_exec_vii:
770 {
771   \tl_use:N \l__enumext_after_list_args_vii_tl
772 }
773 \cs_new_protected:Nn \__enumext_after_args_exec_viii:
774 {
775   \tl_use:N \l__enumext_after_list_args_viii_tl
776 }
```

(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

```

777 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
778 {
779   \keys_define:nn { enumext / #1 }
780   {
781     mini-env .dim_set:c = { \__enumext_minipage_right_#2_dim },
782     mini-env .value_required:n = true,
783     mini-sep .dim_set:c = { \__enumext_minipage_hsep_#2_dim },
784     mini-sep .initial:n = 0.3333em,
785     mini-sep .value_required:n = true,
786     columns-sep .dim_set:c = { \__enumext_columns_sep_#2_dim },
787     columns-sep .value_required:n = true,
788     columns .int_set:c = { \__enumext_columns_#2_int },
789     columns .initial:n = 1,
790     columns .value_required:n = true,
791   }
792 }
793 \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`.

```

794 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
795 {
796   \keys_define:nn { enumext / #1 }
797   {
798     columns .initial:n = 2,
799     miniright .tl_gset:c = { g__enumext_miniright_code_#2_tl },
800     miniright .value_required:n = true,
801     miniright* .code:n = {
802       \bool_gset_true:c { g__enumext_minipage_center_#2_bool }
803       \keys_set:nn { enumext / #1 } { miniright = {##1} }
804     },
805     miniright* .value_required:n = true,
806   }
807 }
808 \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.

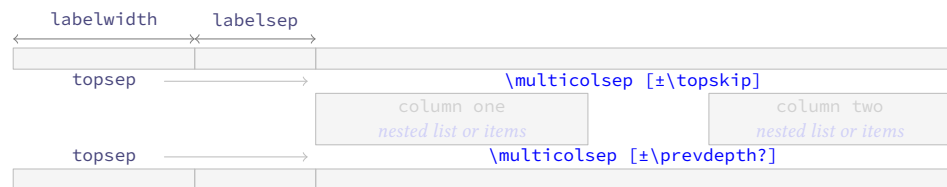


Figure 7: Representation of the vertical space in `multicols` for a nested level.

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 *horizontal mode*, then we will make the settings for the *vertical mode* in which `\partopsep` comes into play.

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

```

809 \cs_new_protected:Nn \__enumext_multi_set_vskip:
810 {
811   \skip_set:cn { \l__enumext_multicols_above_ \__enumext_level: _skip }
812   {
813     \skip_use:c { \l__enumext_topsep_ \__enumext_level: _skip }
814   }
815   \skip_set:cn { \l__enumext_multicols_below_ \__enumext_level: _skip }
816   {
817     \skip_use:c { \l__enumext_topsep_ \__enumext_level: _skip }
818   }
819   \__enumext_add_pre_parsep:
820 }
```

(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*.

```

821 \cs_new_protected:Nn \__enumext_add_pre_parsep:
822 {
823   \int_case:nn { \l__enumext_level_int }
824   {
825     { 2 }{
826       \skip_if_eq:nnF { \l__enumext_parsep_i_skip } { \c_zero_skip }
827       {
828         \skip_add:Nn \l__enumext_multicols_above_ii_skip { \l__enumext_parsep_i_skip }
829       }
830     }
831     { 3 }{
832       \skip_if_eq:nnF { \l__enumext_parsep_ii_skip } { \c_zero_skip }
833       {
834         \skip_add:Nn \l__enumext_multicols_above_iii_skip { \l__enumext_parsep_ii_skip }
835       }
836     }
837     { 4 }{
838       \skip_if_eq:nnF { \l__enumext_parsep_iii_skip } { \c_zero_skip }
839       {
840         \skip_add:Nn \l__enumext_multicols_above_iv_skip { \l__enumext_parsep_iii_skip }
841       }
842     }
843   }
844 }
```

(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 *horizontal mode* or *vertical mode*.

```

845 \cs_new_protected:Nn \__enumext_multi_addvspace:
846 {
847   \__enumext_multi_set_vskip:
848   \mode_if_vertical:T
849   {
850     \skip_add:cn { \l__enumext_multicols_above_ \__enumext_level: _skip }
851     {
852       \skip_use:c { \l__enumext_partopsep_ \__enumext_level: _skip }
853     }
854     \skip_add:cn { \l__enumext_multicols_below_ \__enumext_level: _skip }
855     {
856       \skip_use:c { \l__enumext_partopsep_ \__enumext_level: _skip }
857     }
858   }
```

```

859 \par\nopagebreak
860 \addvspace{ \skip_use:c { \l__enumext_multicols_above_ \l__enumext_level: _skip } }
861 }

```

(End of definition for `__enumext_multi_addvspace:`.)

10.18.2 Adjustment of vertical spaces for multicols in keyans

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

```

862 \cs_new_protected:Nn \__enumext_keyans_multi_set_vskip:
863 {
864   \skip_set:Nn \l__enumext_multicols_above_v_skip
865   {
866     \l__enumext_topsep_v_skip
867   }
868   \skip_set:Nn \l__enumext_multicols_below_v_skip
869   {
870     \l__enumext_topsep_v_skip
871   }
872 }
873 \cs_new_protected:Nn \__enumext_keyans_multi_addvspace:
874 {
875   \__enumext_keyans_multi_set_vskip:
876   \mode_if_vertical:T
877   {
878     \skip_add:Nn \l__enumext_multicols_above_v_skip
879     {
880       \skip_use:N \l__enumext_partopsep_v_skip
881     }
882     \skip_add:Nn \l__enumext_multicols_below_v_skip
883     {
884       \skip_use:N \l__enumext_partopsep_v_skip
885     }
886   }
887   \par\nopagebreak
888   \addvspace{ \l__enumext_multicols_above_v_skip }
889 }

```

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

10.19 Adjustment of vertical spaces for minipage

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

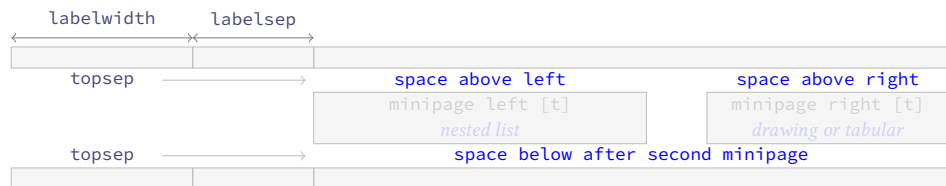


Figure 8: Representation of the `minipage` spacing adjustment for a nested level.

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

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

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

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

```

890 \DeclareDocumentEnvironment{__enumext_mini_env*}{ m }
891 {
892     \__enumext_minipage:w [ t ] { #1 }
893     \legacy_if_gset_false:n { @minipage }
894     \vspace { 0pt }
895 }
896 { \__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`.

```

897 \cs_new_protected:Nn \__enumext_mini_set_vskip:
898 {
899     \int_compare:nNnTF
900     { \int_use:c { \l__enumext_columns_ \__enumext_level: _int } } > { 1 }
901     {

```

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.

```

902     \skip_if_eq:nnTF
903     { \skip_use:c { \l__enumext_topsep_ \__enumext_level: _skip } } { \c_zero_skip }
904     {
905         \skip_set:Nn \l__enumext_minipage_left_skip
906         {
907             -0.150\box_dp:N \strutbox
908         }
909         \skip_set:Nn \l__enumext_minipage_right_skip
910         {
911             0.695\box_dp:N \strutbox
912         }
913         \skip_set:Nn \l__enumext_minipage_after_skip
914         {
915             \box_dp:N \strutbox
916         }
917         \__enumext_zero_parsep:
918     }
919     {
920         \skip_set:Nn \l__enumext_minipage_left_skip
921         {
922             \skip_use:c { \l__enumext_topsep_ \__enumext_level: _skip }
923         }
924         \skip_set:Nn \l__enumext_minipage_right_skip
925         {
926             0.695\box_dp:N \strutbox
927         }
928         \skip_set:Nn \l__enumext_minipage_after_skip
929         {
930             1.85\box_dp:N \strutbox
931             + \skip_use:c { \l__enumext_topsep_ \__enumext_level: _skip }
932         }
933     }
934 }
935 {

```

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.

```

936     \skip_if_eq:nnTF
937     { \skip_use:c { \l__enumext_topsep_ \__enumext_level: _skip } } { \c_zero_skip }
938     {
939         \skip_set:Nn \l__enumext_minipage_left_skip

```

```

940         {
941             0.5\box_dp:N \strutbox
942             - \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
943         }
944     \skip_set:Nn \l__enumext_minipage_right_skip
945     {
946         \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
947     }
948     \skip_set:Nn \l__enumext_minipage_after_skip
949     {
950         1.6\box_dp:N \strutbox
951     }
952 }
953 {
954     \skip_set:Nn \l__enumext_minipage_left_skip
955     {
956         0.5875\box_dp:N \strutbox
957         - \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
958     }
959     \skip_set:Nn \l__enumext_minipage_right_skip
960     {
961         + \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
962         + \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
963     }
964     \skip_set:Nn \l__enumext_minipage_after_skip
965     {
966         0.325\box_dp:N \strutbox
967         + \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
968     }
969 }
970 }
971 }

```

(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.

```

972 \cs_new_protected:Nn \__enumext_zero_parsep:
973 {
974     \int_case:nn { \l__enumext_level_int }
975     {
976         { 2 }{
977             \skip_if_eq:nnF { \l__enumext_parsep_i_skip } { \c_zero_skip }
978             {
979                 \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
980             }
981         }
982         { 3 }{
983             \skip_if_eq:nnF { \l__enumext_parsep_ii_skip } { \c_zero_skip }
984             {
985                 \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
986             }
987         }
988         { 4 }{
989             \skip_if_eq:nnF { \l__enumext_parsep_iii_skip } { \c_zero_skip }
990             {
991                 \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
992             }
993         }
994     }
995 }

```

(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*.

```

996 \cs_new_protected:Nn \__enumext_mini_addvspace:
997 {

```

```

998   \__enumext_mini_set_vskip:
999   \mode_if_vertical:T
1000   {
1001     \skip_add:Nn \l__enumext_minipage_left_skip
1002     {
1003       \skip_use:c { \l__enumext_partopsep_ \__enumext_level: _skip }
1004     }
1005     \skip_add:Nn \l__enumext_minipage_after_skip
1006     {
1007       \skip_use:c { \l__enumext_partopsep_ \__enumext_level: _skip }
1008     }
1009   }
1010   \par\nopagebreak
1011   \addvspace { \l__enumext_minipage_left_skip }
1012 }

```

(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`.

```

1013 \cs_new_protected:Nn \__enumext_keyans_mini_set_vskip:
1014 {
1015   \skip_zero_new:N \l__enumext_minipage_after_skip
1016   \skip_zero_new:N \l__enumext_minipage_left_skip
1017   \skip_zero_new:N \l__enumext_minipage_right_skip
1018   \int_compare:nNnTF { \l__enumext_columns_v_int } > { 1 }
1019   {
1020     \skip_if_eq:nnTF { \l__enumext_topsep_v_skip } { \c_zero_skip }
1021     {
1022       \skip_set:Nn \l__enumext_minipage_left_skip { -0.25\box_dp:N \strutbox }
1023       \skip_set:Nn \l__enumext_minipage_right_skip { 0.705\box_dp:N \strutbox }
1024       \skip_set:Nn \l__enumext_minipage_after_skip { \box_dp:N \strutbox }
1025       \skip_if_eq:nnF { \l__enumext_parsep_i_skip } { \c_zero_skip }
1026       {
1027         \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
1028       }
1029     }
1030     {
1031       \skip_set:Nn \l__enumext_minipage_left_skip
1032       {
1033         \skip_use:N \l__enumext_topsep_v_skip
1034       }
1035       \skip_set:Nn \l__enumext_minipage_right_skip
1036       {
1037         0.705\box_dp:N \strutbox
1038       }
1039       \skip_set:Nn \l__enumext_minipage_after_skip
1040       {
1041         1.85\box_dp:N \strutbox + \l__enumext_topsep_v_skip
1042       }
1043     }
1044   }
1045   {
1046     \skip_if_eq:nnTF { \l__enumext_topsep_v_skip } { \c_zero_skip }
1047     {
1048       \skip_set:Nn \l__enumext_minipage_left_skip
1049       {
1050         0.5\box_dp:N \strutbox
1051         + \l__enumext_partopsep_v_skip
1052       }
1053       \skip_set:Nn \l__enumext_minipage_right_skip
1054       {
1055         \l__enumext_partopsep_v_skip
1056       }
1057       \skip_set:Nn \l__enumext_minipage_after_skip { 1.6\box_dp:N \strutbox }
1058     }
1059     {
1060       \skip_set:Nn \l__enumext_minipage_left_skip
1061       {

```

```

1062         0.5875\box_dp:N \strutbox - \l__enumext_partopsep_v_skip
1063     }
1064     \skip_set:Nn \l__enumext_minipage_right_skip
1065     {
1066         \l__enumext_topsep_v_skip + \l__enumext_partopsep_v_skip
1067     }
1068     \skip_set:Nn \l__enumext_minipage_after_skip
1069     {
1070         0.325\box_dp:N \strutbox + \l__enumext_topsep_v_skip
1071     }
1072 }
1073 }
1074 }

```

(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`.

```

1075 \cs_new_protected:Nn \__enumext_keyans_mini_addvspace:
1076 {
1077     \__enumext_keyans_mini_set_vskip:
1078     \mode_if_vertical:T
1079     {
1080         \skip_add:Nn \l__enumext_minipage_left_skip
1081         {
1082             \l__enumext_partopsep_v_skip
1083         }
1084         \skip_add:Nn \l__enumext_minipage_after_skip
1085         {
1086             \l__enumext_partopsep_v_skip
1087         }
1088     }
1089     \par\nopagebreak
1090     \addvspace { \l__enumext_minipage_left_skip }
1091 }

```

(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*`.

```

1092 \cs_new_protected:Nn \__enumext_mini_set_vskip_vii:
1093 {
1094     \skip_zero_new:N \l__enumext_minipage_left_skip
1095     \skip_gzero_new:N \g__enumext_minipage_right_skip
1096     \skip_gzero_new:N \g__enumext_minipage_after_skip
1097     \skip_if_eq:nnTF { \l__enumext_topsep_vii_skip } { \c_zero_skip }
1098     {
1099         \skip_set:Nn \l__enumext_minipage_left_skip { 0.5\box_dp:N \strutbox }
1100         \skip_gset:Nn \g__enumext_minipage_right_skip { 0.325\box_dp:N \strutbox }
1101     }
1102     {
1103         \skip_set:Nn \l__enumext_minipage_left_skip { 0.5875\box_dp:N \strutbox }
1104         \skip_gset:Nn \g__enumext_minipage_right_skip
1105         {
1106             \l__enumext_topsep_vii_skip
1107         }
1108         \skip_gset:Nn \g__enumext_minipage_after_skip
1109         {
1110             0.325\box_dp:N \strutbox + \l__enumext_topsep_vii_skip
1111         }
1112     }
1113 }
1114 \cs_new_protected:Nn \__enumext_mini_set_vskip_viii:
1115 {
1116     \skip_zero_new:N \l__enumext_minipage_after_skip

```

```

1117 \skip_zero_new:N \l__enumext_minipage_left_skip
1118 \skip_zero_new:N \l__enumext_minipage_right_skip
1119 \skip_if_eq:nnTF { \l__enumext_topsep_viii_skip } { \c_zero_skip }
1120 {
1121   \skip_set:Nn \l__enumext_minipage_left_skip
1122   {
1123     0.5\box_dp:N \strutbox
1124   }
1125   \skip_set:Nn \l__enumext_minipage_right_skip
1126   {
1127     \l__enumext_partopsep_viii_skip
1128   }
1129   \skip_set:Nn \l__enumext_minipage_after_skip
1130   {
1131     1.6\box_dp:N \strutbox
1132   }
1133 }
1134 {
1135   \skip_set:Nn \l__enumext_minipage_left_skip
1136   {
1137     0.5875\box_dp:N \strutbox
1138   }
1139   \skip_set:Nn \l__enumext_minipage_right_skip
1140   {
1141     \l__enumext_topsep_viii_skip
1142   }
1143   \skip_set:Nn \l__enumext_minipage_after_skip
1144   {
1145     0.325\box_dp:N \strutbox + \l__enumext_topsep_viii_skip
1146   }
1147 }
1148 }

```

(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.

```

1149 \cs_new_protected:Nn \__enumext_mini_addvspace_vii:
1150 {
1151   \__enumext_mini_set_vskip_vii:
1152   \par\nopagebreak
1153   \addvspace { \l__enumext_minipage_left_skip }
1154 }
1155 \cs_new_protected:Nn \__enumext_mini_addvspace_viii:
1156 {
1157   \__enumext_mini_set_vskip_viii:
1158   \par\nopagebreak
1159   \addvspace { \l__enumext_minipage_left_skip }
1160 }

```

(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.

```

1161 \NewDocumentCommand \miniright { s }
1162 {
1163   \int_compare:nNt { \l__enumext_keyans_pic_level_int } = { 1 }
1164   {
1165     \msg_error:nnn { enumext } { wrong-miniright-place }

```



```

1166     }
1167     \int_compare:nNt { \__enumext_level_int } = { 0 }
1168     {
1169         \msg_error:nnn { enumext } { wrong-miniright-place }
1170     }
1171     \int_compare:nNtF { \__enumext_keyans_level_int } = { 1 }
1172     {
1173         \__enumext_keyans_mini_right_cmd:n {#1}
1174     }
1175     { \__enumext_mini_right_cmd:n {#1} }
1176 }

```

(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.

```

1177 \cs_new_protected:Npn \__enumext_mini_right_cmd:n #1
1178 {
1179     \dim_compare:nNtF
1180     { \dim_use:c { \__enumext_minipage_right_ \__enumext_level: _dim } } > { \c_zero_dim }
1181     {
1182         \__enumext_multicols_stop:
1183         \end{__enumext_mini_env*}
1184         \hfill
1185         \begin{__enumext_mini_env*}
1186         { \dim_use:c { \__enumext_minipage_right_ \__enumext_level: _dim } }
1187         \par\addvspace { \__enumext_minipage_right_skip }
1188         \bool_if:nF {#1}
1189         {
1190             \centering
1191         }
1192         \int_gzero:N \g__enumext_minipage_stat_int
1193     }
1194     { \msg_error:nnn { enumext } { wrong-miniright-use } }
1195 }

```

(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.

```

1196 \cs_new_protected:Npn \__enumext_keyans_mini_right_cmd:n #1
1197 {
1198     \dim_compare:nNtF { \__enumext_minipage_right_v_dim } > { \c_zero_dim }
1199     {
1200         \__enumext_keyans_multicols_stop:
1201         \end{__enumext_mini_env*}
1202         \hfill
1203         \begin{__enumext_mini_env*}{ \__enumext_minipage_right_v_dim }
1204         \par\addvspace { \__enumext_minipage_right_skip }
1205         \bool_if:nF {#1}
1206         {
1207             \centering
1208         }
1209         \int_gzero:N \g__enumext_minipage_stat_int
1210     }
1211     { \msg_error:nnn { enumext } { wrong-miniright-use } }
1212 }

```

(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.

```

1213 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1214 {
1215   \keys_define:nn { enumext / #1 }
1216   {
1217     above .skip_set:c = { \__enumext_vspace_above_#2_skip },
1218     above .value_required:n = true,
1219     above* .code:n      = \bool_set_true:c { \__enumext_vspace_a_star_#2_bool }
1220                      \keys_set:nn { enumext / #1 } { above = {##1} },
1221     above* .value_required:n = true,
1222     below .skip_set:c = { \__enumext_vspace_below_#2_skip },
1223     below .value_required:n = true,
1224     below* .code:n      = \bool_set_true:c { \__enumext_vspace_b_star_#2_bool }
1225                      \keys_set:nn { enumext / #1 } { below = {##1} },
1226     below* .value_required:n = true,
1227   }
1228 }
1229 \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.

```

1230 \cs_new_protected:Nn \__enumext_vspace_above:
1231 {
1232   \skip_if_eq:nnF
1233   { \skip_use:c { \__enumext_vspace_above_ \__enumext_level: _skip } } { \c_zero_skip }
1234   {
1235     \bool_if:cTF { \__enumext_vspace_a_star_ \__enumext_level: _bool }
1236     {
1237       \vspace*{ \skip_use:c { \__enumext_vspace_above_ \__enumext_level: _skip } }
1238     }
1239     {
1240       \vspace { \skip_use:c { \__enumext_vspace_above_ \__enumext_level: _skip } }
1241     }
1242   }
1243 }

```

(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.

```

1244 \cs_new_protected:Nn \__enumext_vspace_below:
1245 {
1246   \skip_if_eq:nnF
1247   { \skip_use:c { \__enumext_vspace_below_ \__enumext_level: _skip } } { \c_zero_skip }
1248   {
1249     \bool_if:cTF { \__enumext_vspace_b_star_ \__enumext_level: _bool }
1250     {
1251       \vspace*{ \skip_use:c { \__enumext_vspace_below_ \__enumext_level: _skip } }
1252     }
1253     {
1254       \vspace { \skip_use:c { \__enumext_vspace_below_ \__enumext_level: _skip } }
1255     }
1256   }
1257 }

```

(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.

```

1258 \cs_new_protected:Nn \__enumext_vspace_above_v:
1259 {
1260   \skip_if_eq:nnF { \l__enumext_vspace_above_v_skip } { \c_zero_skip }
1261   {
1262     \bool_if:NTF \l__enumext_vspace_a_star_v_bool
1263     {
1264       \vspace*{ \l__enumext_vspace_above_v_skip }
1265     }
1266     { \vspace { \l__enumext_vspace_above_v_skip } }
1267   }
1268 }
```

(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.

```

1269 \cs_new_protected:Nn \__enumext_vspace_below_v:
1270 {
1271   \skip_if_eq:nnF { \l__enumext_vspace_below_v_skip } { \c_zero_skip }
1272   {
1273     \bool_if:NTF \l__enumext_vspace_b_star_v_bool
1274     {
1275       \vspace*{ \l__enumext_vspace_below_v_skip }
1276     }
1277     { \vspace { \l__enumext_vspace_below_v_skip } }
1278   }
1279 }
```

(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:`

```

1280 \cs_new_protected:Nn \__enumext_vspace_above_vii:
1281 {
1282   \skip_if_eq:nnF { \l__enumext_vspace_above_vii_skip } { \c_zero_skip }
1283   {
1284     \bool_if:NTF \l__enumext_vspace_a_star_vii_bool
1285     {
1286       \vspace*{ \l__enumext_vspace_above_vii_skip }
1287     }
1288     { \vspace { \l__enumext_vspace_above_vii_skip } }
1289   }
1290 }
1291 \cs_new_protected:Nn \__enumext_vspace_above_viii:
1292 {
1293   \skip_if_eq:nnF { \l__enumext_vspace_above_viii_skip } { \c_zero_skip }
1294   {
1295     \bool_if:NTF \l__enumext_vspace_a_star_viii_bool
1296     {
1297       \vspace*{ \l__enumext_vspace_above_viii_skip }
1298     }
1299     { \vspace { \l__enumext_vspace_above_viii_skip } }
1300   }
1301 }
```

(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:`

```

1302 \cs_new_protected:Nn \__enumext_vspace_below_vii:
1303 {
1304   \skip_if_eq:nnF { \l__enumext_vspace_below_vii_skip } { \c_zero_skip }
1305   {
1306     \bool_if:NTF \l__enumext_vspace_b_star_vii_bool
1307     {
1308       \vspace*{ \l__enumext_vspace_below_vii_skip }

```

```

1309     }
1310     { \vspace { \l__enumext_vspace_below_vii_skip } }
1311   }
1312 }
1313 \cs_new_protected:Nn \__enumext_vspace_below_viii:
1314 {
1315   \skip_if_eq:nnF { \l__enumext_vspace_below_viii_skip } { \c_zero_skip }
1316   {
1317     \bool_if:NTF \l__enumext_vspace_b_star_viii_bool
1318     {
1319       \vspace*{ \l__enumext_vspace_below_viii_skip }
1320     }
1321     { \vspace { \l__enumext_vspace_below_viii_skip } }
1322   }
1323 }

```

(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*
1324 \keys_define:nn { enumext / level-1 }
1325 {
1326   save-ans .code:n = \__enumext_storing_set:n {#1},
1327   save-ans .value_required:n = true,
1328   resume .code:n = \__enumext_resume_counter:,
1329   resume .value_forbidden:n = true,
1330   resume* .code:n = \__enumext_resume_counter_star:,
1331   resume* .value_forbidden:n = true,
1332 }
1333 \keys_define:nn { enumext / enumext* }
1334 {
1335   save-ans .code:n = \__enumext_storing_set:n {#1},
1336   save-ans .value_required:n = true,
1337   resume .code:n = \__enumext_resume_counter_vii:,
1338   resume .value_forbidden:n = true,
1339 }

```

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

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.

```

1340 \cs_new_protected:Npn \__enumext_storing_set:n #1
1341 {
1342   \tl_set:Nx \l__enumext_store_name_tl {#1}
1343   \prop_if_exist:cF { g__enumext_ \l__enumext_store_name_tl _prop }
1344   {
1345     \prop_new:c { g__enumext_ \l__enumext_store_name_tl _prop }
1346   }
1347   \seq_if_exist:cF { g__enumext_ \l__enumext_store_name_tl _seq }
1348   {
1349     \seq_new:c { g__enumext_ \l__enumext_store_name_tl _seq }
1350   }
1351   \bool_set_true:N \l__enumext_store_active_bool
1352   \int_if_exist:cF { g__enumext_resume_#1_int }
1353   {
1354     \int_new:c { g__enumext_resume_#1_int }
1355   }
1356   \bool_if:NT \l__enumext_check_ans_bool
1357   {
1358     \__enumext_check_ans_int:n {#1}
1359   }
1360 }

```

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

`__enumext_resume_counter:` 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`.

```

1361 \cs_new_protected:Nn \__enumext_resume_counter:
1362 {
1363   \bool_if:NT \l__enumext_store_active_bool
1364   {
1365     \int_gset:Nn \g__enumext_resume_int
1366     {
1367       \int_use:c { g__enumext_resume_ \l__enumext_store_name_tl _int }
1368     }
1369   }
1370   \bool_set_true:N \l__enumext_resume_bool
1371 }
1372 \cs_new_protected:Nn \__enumext_resume_counter_vii:
1373 {
1374   \bool_if:NT \l__enumext_store_active_bool
1375   {
1376     \int_gset:Nn \g__enumext_resume_int
1377     {
1378       \int_use:c { g__enumext_resume_ \l__enumext_store_name_tl _int }
1379     }
1380   }
1381   \bool_set_true:N \l__enumext_resume_vii_bool
1382 }

```

(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 1383 \cs_set_protected:Npn \__enumext_tmp:n #1
no-store 1384 {
1385   \keys_define:nn { enumext / #1 }
1386   {
1387     check-ans .bool_set:N = \l__enumext_check_ans_bool,
1388     check-ans .initial:n = false,
1389     no-store .code:n = {
1390       \bool_set_false:N \l__enumext_store_ans_bool
1391       \bool_set_false:N \l__enumext_check_ans_bool
1392     },
1393     no-store .value_forbidden:n = true,
1394   }
1395 }
1396 \clist_map_inline:nn
1397 {
1398   level-1, level-2, level-3, level-4, enumext*
1399 }
1400 { \__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`

```

1401 \cs_new_protected:Npn \__enumext_check_ans_int:n #1
1402 {
1403   \int_if_exist:cF { g__enumext_count_#1_item_ans_int }
1404   { \int_new:c { g__enumext_count_#1_item_ans_int } }
1405   \int_if_exist:cF { g__enumext_count_#1_i_int }
1406   { \int_new:c { g__enumext_count_#1_i_int } }
1407   \int_if_exist:cF { g__enumext_count_#1_ii_int }
1408   { \int_new:c { g__enumext_count_#1_ii_int } }
1409   \int_if_exist:cF { g__enumext_count_#1_iii_int }
1410   { \int_new:c { g__enumext_count_#1_iii_int } }
1411   \int_if_exist:cF { g__enumext_count_#1_iv_int }
1412   { \int_new:c { g__enumext_count_#1_iv_int } }
1413   \int_if_exist:cF { g__enumext_count_#1_vii_int }
1414   { \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.

```

1415   \bool_lazy_all:nTF
1416   {
1417     { \g__enumext_starred_bool }
1418     { \int_compare_p:nNn { \l__enumext_level_int } = { \c_zero_int } }
1419   }
1420   {
1421     \int_gset_eq:Nc \g__enumext_count_item_all_int { g__enumext_count_#1_vii_int }
1422   }
1423   {
1424     \int_gset_eq:Nc \g__enumext_count_item_all_int { g__enumext_count_#1_i_int }
1425   }
1426   \int_gset_eq:Nc \g__enumext_count_item_i_int { g__enumext_count_#1_i_int }
1427   \int_gset_eq:Nc \g__enumext_count_item_ii_int { g__enumext_count_#1_ii_int }
1428   \int_gset_eq:Nc \g__enumext_count_item_iii_int { g__enumext_count_#1_iii_int }
1429   \int_gset_eq:Nc \g__enumext_count_item_iv_int { g__enumext_count_#1_iv_int }
1430   \int_gset_eq:Nc \g__enumext_count_item_vii_int { g__enumext_count_#1_vii_int }
1431   \int_gset_eq:Nc \g__enumext_count_item_with_ans_int { g__enumext_count_#1_item_ans_int }
1432 }
```

(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.

```

1433 \cs_new_protected:Nn \__enumext_check_ans_count:
1434 {
1435   \bool_if:NT \l__enumext_check_ans_bool
1436   {
1437     \bool_if:NTF \l__enumext_store_ans_bool
1438     {
1439       \int_gadd:cn { g__enumext_count_item_ \__enumext_level: _int }
1440       { \int_use:c { g__enumext_count_level_ \__enumext_level: _int } + 1 }
1441     }
1442     { \int_gincr:c { g__enumext_count_item_ \__enumext_level: _int } }
1443   }
1444 }

```

(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:`

```

1445 \cs_new_protected:Nn \__enumext_check_ans_active:
1446 {
1447   \int_set:Nn \l__enumext_compare_items_ans_int
1448   {
1449     \g__enumext_count_item_all_int - \g__enumext_count_item_ii_int
1450     - \g__enumext_count_item_iii_int - \g__enumext_count_item_iv_int
1451   }
1452   \int_compare:nNnTF
1453   { \l__enumext_compare_items_ans_int } = { \g__enumext_count_item_with_ans_int }
1454   {
1455     \msg_term:nnV { enumext } { items-same-answer } \g__enumext_store_name_tl
1456   }
1457   {
1458     \msg_warning:nnV { enumext } { item-different-answer } \g__enumext_store_name_tl
1459   }

```

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

```

1460   \__enumext_zero_count_level:
1461 }

1462 \cs_new_protected:Nn \__enumext_check_ans_active_vii:
1463 {
1464   \int_set:Nn \l__enumext_compare_items_ans_int
1465   {
1466     \g__enumext_count_item_all_int - \g__enumext_count_item_i_int
1467     - \g__enumext_count_item_ii_int - \g__enumext_count_item_iii_int
1468     - \g__enumext_count_item_iv_int
1469   }
1470   \int_compare:nNnTF
1471   { \l__enumext_compare_items_ans_int } = { \g__enumext_count_item_with_ans_int }
1472   {
1473     \msg_term:nnV { enumext } { items-same-answer } \g__enumext_store_name_tl
1474   }
1475   {
1476     \msg_warning:nnV { enumext } { item-different-answer } \g__enumext_store_name_tl
1477   }
1478   \__enumext_zero_count_level:
1479 }

```

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

10.23 Keys and functions associated with storage

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

```

1480 \cs_set_protected:Npn \__enumext_tmp:n #1
1481 {
1482   \keys_define:nn { enumext / #1 }

```



```

1483     {
1484         wrap-ans .cs_set_protected:Np = \__enumext_anskey_wrapper:n #1,
1485         wrap-ans .initial:n = \fbox{##1},
1486         wrap-ans .value_required:n = true,
1487         mark-ans .code:n = \tl_set:Nn \__enumext_mark_answer_sym_tl {##1},
1488         mark-ans .initial:n = \textasteriskcentered,
1489         mark-ans .value_required:n = true,
1490         mark-pos .choice:,
1491         mark-pos / left .code:n = \str_set:Nn \__enumext_mark_position_str { l },
1492         mark-pos / right .code:n = \str_set:Nn \__enumext_mark_position_str { r },
1493         mark-pos .initial:n = right,
1494         mark-pos .value_required:n = true,
1495         show-ans .code:n = \bool_set_true:N \__enumext_show_answer_bool
1496                     \bool_set_false:N \__enumext_show_position_bool,
1497         show-ans .value_forbidden:n = true,
1498         show-pos .code:n = \bool_set_true:N \__enumext_show_position_bool
1499                     \bool_set_false:N \__enumext_show_answer_bool,
1500         show-pos .value_forbidden:n = true,
1501         mark-ref .code:n = \tl_set:Nn \__enumext_mark_ref_sym_tl {##1},
1502         mark-ref .initial:n = \textasteriskcentered,
1503         mark-ref .value_required:n = true,
1504         store-ref .bool_set:N = \__enumext_store_ref_key_bool,
1505         store-ref .initial:n = false,
1506     }
1507 }
1508 \clist_map_inline:nn { level-1, enumext* } { \__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-
show-ans pos.

```

1509 \cs_set_protected:Npn \__enumext_tmp:n #1
1510 {
1511     \keys_define:nn { enumext / #1 }
1512     {
1513         mark-pos .choice:,
1514         mark-pos / left .code:n = \str_set:Nn \__enumext_mark_position_str { l },
1515         mark-pos / right .code:n = \str_set:Nn \__enumext_mark_position_str { r },
1516         mark-pos .initial:n = right,
1517         mark-pos .value_required:n = true,
1518         show-ans .code:n = \bool_set_true:N \__enumext_show_answer_bool
1519                     \bool_set_false:N \__enumext_show_position_bool,
1520         show-ans .value_forbidden:n = true,
1521         show-pos .code:n = \bool_set_true:N \__enumext_show_position_bool
1522                     \bool_set_false:N \__enumext_show_answer_bool,
1523         show-pos .value_forbidden:n = true,
1524     }
1525 }
1526 \clist_map_inline:nn { keyans, keyans* } { \__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 `__enumext_store_level_open:` function used by the “storage system” to preserve the structure and then used by the `\printkeyans` command.

```

1527 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1528 {
1529     \keys_define:nn { enumext / #1 }
1530     {
1531         columns* .code:n = \bool_set_true:c { \__enumext_store_columns_#2_bool }
1532                     \int_set:cn { \__enumext_store_columns_#2_int } {##1}
1533                     \tl_put_right:ce { \__enumext_store_opt_#2_tl }
1534                     {
1535                         columns = \exp_not:v { \__enumext_store_columns_#2_int },
1536                     },,
1537         columns* .value_required:n = true,
1538         columns-sep* .code:n = \bool_set_true:c { \__enumext_store_columns_sep_#2_bool }
1539                     \dim_set:cn { \__enumext_store_columns_sep_#2_dim } {##1}
1540                     \tl_put_right:ce { \__enumext_store_opt_#2_tl }
1541                     {

```

```

1542             columns-sep = \exp_not:v { l__enumext_store_columns_sep_#2_dir
1543             },
1544             columns-sep* .value_required:n = true,
1545         }
1546     }
1547 \clist_map_inline:nn
1548 {
1549     {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {enumext*}{vii}
1550 }
1551 { \__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` The function `__enumext_store_addto_prop:n` stores the content in *prop list* defined by `save-ans` key. The “*stored content*” is retrieved by means of the `\getkeyans` command.

`__enumext_store_addto_prop:V`

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

```

1552 \cs_generate_variant:Nn \prop_gput_if_not_in:Nnn { cen }
1553 \cs_new_protected:Npn \__enumext_store_addto_prop:n #1
1554 {
1555     \prop_gput_if_not_in:cen { g__enumext_ \l__enumext_store_name_tl _prop }
1556     {
1557         \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } + \c_one_int }
1558     }
1559     { #1 }
1560 }
1561 \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` The function `__enumext_store_addto_seq:n` stores the content in *sequence* defined by `save-ans` key. This function is used by `\anskey` in `enumext`, `\item*` in `keyans` and `\anspic` in `keyanspic`.

`__enumext_store_addto_seq:v`

`__enumext_store_addto_seq:V` The form in which the content is stored in *sequence* is in a internal `enumext` or `enumext*` environments with the *same structure* in which the command was executed.

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

```

1562 \cs_new_protected:Npn \__enumext_store_addto_seq:n #1
1563 {
1564     \seq_gput_right:cn { g__enumext_ \l__enumext_store_name_tl _seq } { #1 }
1565 }
1566 \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:` 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`.

`__enumext_store_level_close:`

```

1567 \cs_new_protected:Nn \__enumext_store_level_open:
1568 {
1569     \bool_if:NF \l__enumext_store_ans_bool
1570     {
1571         \tl_if_empty:cTF { l__enumext_store_opt_ \l__enumext_level: _tl }
1572         {
1573             \__enumext_store_addto_seq:n
1574             {
1575                 \item \begin{enumext}
1576             }
1577         }
1578         {
1579             \tl_put_left:cn { l__enumext_store_opt_ \l__enumext_level: _tl }
1580             {
1581                 \item \begin{enumext} [

```

```

1582         }
1583         \tl_put_right:cn { \__enumext_store_opt_ \__enumext_level: _tl }
1584         {
1585             ]
1586         }
1587         \__enumext_store_addto_seq:v { \__enumext_store_opt_ \__enumext_level: _tl }
1588     }
1589 }
1590 }
1591 \cs_new_protected:Nn \__enumext_store_level_close:
1592 {
1593     \bool_if:NF \l__enumext_store_ans_bool
1594     {
1595         \__enumext_store_addto_seq:n { \end{enumext} }
1596     }
1597 }

```

(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.

```

1598 \cs_new_protected:Nn \__enumext_store_level_open_vii:
1599 {
1600     \bool_if:NF \l__enumext_store_ans_bool
1601     {
1602         \tl_if_empty:NTF \l__enumext_store_opt_vii_tl
1603         {
1604             \__enumext_store_addto_seq:n
1605             {
1606                 \item \mode_leave_vertical:
1607                 \vspace { -\skip_eval:n { \baselineskip + \parsep } }
1608                 \begin{enumext*}[before={\setlength{\topsep}{\opt}},]
1609             }
1610         }
1611         {
1612             \tl_put_left:Nn \l__enumext_store_opt_vii_tl
1613             {
1614                 \item \mode_leave_vertical:
1615                 \vspace { -\skip_eval:n { \baselineskip + \parsep } }
1616                 \begin{enumext*}[before={\setlength{\topsep}{\opt}},
1617                 ]
1618                 \tl_put_right:Nn \l__enumext_store_opt_vii_tl
1619                 {
1620                     ]
1621                 }
1622                 \__enumext_store_addto_seq:V \l__enumext_store_opt_vii_tl
1623             }
1624         }
1625     }
1626 \cs_new_protected:Nn \__enumext_store_level_close_vii:
1627 {
1628     \bool_if:NF \l__enumext_store_ans_bool
1629     {
1630         \__enumext_store_addto_seq:n { \end{enumext*} }
1631     }
1632 }

```

(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`

```

1633 \cs_new_protected:Nn \__enumext_print_keyans_box:NN
1634 {
1635   \mode_leave_vertical:
1636   \skip_horizontal:n { -\dim_use:N #2 }
1637   \makebox[0pt][ r ]
1638   {
1639     \makebox[ \dim_use:N #1 ][ \__enumext_mark_position_str ]
1640     {
1641       \tl_use:N \__enumext_mark_answer_sym_tl
1642     }
1643   }
1644   \skip_horizontal:n { \dim_use:N #2 }
1645 }
1646 \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`.

```

1647 \keys_define:nn { enumext / anskey }
1648 {
1649   item-sym* .tl_set:N = \__enumext_store_item_symbol_tl,
1650   item-sym* .value_required:n = true,
1651   item-pos* .dim_set:N = \__enumext_store_item_symbol_sep_dim,
1652   item-pos* .value_required:n = true,
1653   store-brk .bool_set:N = \__enumext_store_columns_break_bool,
1654   store-brk .default:n = true,
1655   store-brk .value_forbidden:n = true,
1656 }

```

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)*.

```

1657 \NewDocumentCommand \anskey { d() s o +m }
1658 {
1659   \bool_if:NF \__enumext_store_active_bool
1660   {
1661     \msg_error:nnnn { enumext } { anskey-wrong-place } { anskey } { enumext }
1662   }
1663   \int_compare:nNnT { \__enumext_keyans_level_int } = { 1 }
1664   {
1665     \msg_error:nnnn { enumext } { command-wrong-place } { anskey } { keyans }
1666   }
1667   \int_compare:nNnT { \__enumext_keyans_pic_level_int } = { 1 }
1668   {
1669     \msg_error:nnnn { enumext } { command-wrong-place } { anskey } { keyanspic }
1670   }
1671   \group_begin:
1672     \bool_if:NF \__enumext_store_ans_bool
1673     {
1674       \bool_if:NT \__enumext_check_ans_bool
1675       {
1676         \int_gincr:N \g__enumext_count_item_with_ans_int
1677       }
1678       \__enumext_store_anskey_code:nnnn {#1} {#2} {#3} {#4}
1679     }
1680   \group_end:
1681 }

```

(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 `__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.

```

1682 \cs_new_protected:Npn \__enumext_store_anskey_code:nnnn #1 #2 #3 #4
1683 {
1684   \__enumext_store_addto_prop:n {#4}
1685   \bool_if:NT \l__enumext_store_ref_key_bool
1686   {
1687     \__enumext_store_internal_ref:
1688   }
1689   \__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`.

```

1690   \tl_clear:N \l__enumext_store_anskey_arg_tl
1691   \tl_if_novalue:nF {#3}
1692   {
1693     \keys_set:nn { enumext / anskey } {#3}
1694   }
1695   \bool_lazy_and:nnT
1696   { \l__enumext_store_columns_break_bool }
1697   { \bool_not_p:n { \l__enumext_starred_bool } }
1698   {
1699     \tl_put_left:Nn \l__enumext_store_anskey_arg_tl { \columnbreak }
1700   }
1701   \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).

```

1702   \tl_if_novalue:nF {#1}
1703   {
1704     \int_set:Nn \l__enumext_store_columns_join_int {#1}
1705     \bool_if:NT \l__enumext_starred_bool
1706     {
1707       \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
1708       {
1709         ( \exp_not:V \l__enumext_store_columns_join_int )
1710       }
1711     }
1712   }

```

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`.

```

1713   \bool_if:nTF {#2}
1714   {
1715     \tl_put_right:Nn \l__enumext_store_anskey_arg_tl { * }
1716     \tl_if_empty:NF \l__enumext_store_item_symbol_tl
1717     {
1718       \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
1719       {
1720         [ \exp_not:V \l__enumext_store_item_symbol_tl ]
1721       }
1722     }
1723     \dim_compare:nT
1724     {
1725       \l__enumext_store_item_symbol_sep_dim != \c_zero_dim
1726     }
1727     {
1728       \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
1729       {
1730         [ \exp_not:V \l__enumext_store_item_symbol_sep_dim ]
1731       }
1732     }
1733     \tl_put_right:Nn \l__enumext_store_anskey_arg_tl {#4}
1734   }
1735   {
1736     \tl_put_right:Nn \l__enumext_store_anskey_arg_tl {#4}
1737   }

```

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`.

```

1738 \bool_lazy_and:nnT
1739 { \l__enumext_store_ref_key_bool }
1740 { \l__enumext_hyperref_bool }
1741 {
1742   \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
1743   {
1744     \hfill \exp_not:N \hyperlink { \exp_not:V \l__enumext_newlabel_arg_one_tl }
1745     { \exp_not:V \l__enumext_mark_ref_sym_tl }
1746   }
1747 }
1748 \__enumext_store_addto_seq:V \l__enumext_store_anskey_arg_tl
1749 }

```

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

`__enumext_store_internal_ref:`

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

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

```

1750 \cs_new_protected:Nn \__enumext_store_internal_ref:
1751 {
1752   \cs_set_protected:Npn \__enumext_tmp:n ##1
1753   {
1754     \tl_set_eq:cc { \l__enumext_label_copy_##1_tl } { \l__enumext_label_##1_tl }
1755     \tl_reverse:c { \l__enumext_label_copy_##1_tl }
1756     \tl_remove_once:cn { \l__enumext_label_copy_##1_tl } { . }
1757     \tl_reverse:c { \l__enumext_label_copy_##1_tl }
1758   }
1759   \clist_map_inline:nn { i, ii, iii, iv, vii } { \__enumext_tmp:n {##1} }
1760   \cs_set:Npn \__enumext_tmp:n ##1
1761   { . \tl_use:c { \l__enumext_label_copy_ \int_to_roman:n {##1} _tl } }

```

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

```

1762 \bool_lazy_all:nT
1763 {
1764   { \g__enumext_starred_bool }
1765   { \int_compare_p:nNn { \l__enumext_level_int } = { \c_zero_int } }
1766 }
1767 {
1768   \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
1769   { \tl_use:N \l__enumext_label_copy_vii_tl }
1770 }
1771 \bool_lazy_all:nT
1772 {
1773   { \l__enumext_standar_bool }
1774   { \g__enumext_starred_bool }
1775   { \int_compare_p:nNn { \l__enumext_level_int } > { \c_zero_int } }
1776 }
1777 {
1778   \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
1779   {
1780     \tl_use:N \l__enumext_label_copy_vii_tl
1781     \int_step_function:nnN { 1 } { \l__enumext_level_int } \__enumext_tmp:n
1782   }
1783 }

```

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.

```

1784 \bool_lazy_all:nT
1785 {
1786   { \l__enumext_standar_bool }
1787   { \int_compare_p:nNn { \l__enumext_level_int } > { \c_zero_int } }
1788   { \int_compare_p:nNn { \l__enumext_level_h_int } = { \c_zero_int } }
1789   { \bool_not_p:n { \l__enumext_starred_bool } }
1790 }
1791 {
1792   \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl

```

```

1793     {
1794         \tl_use:N \l__enumext_label_copy_i_tl
1795         \int_step_function:nnN { 2 } { \l__enumext_level_int } \__enumext_tmp:n
1796     }
1797 }
1798 \cs_set:Npn \__enumext_tmp:n ##1
1799 { \tl_use:c { l__enumext_label_copy_ \int_to_roman:n {##1} _tl } }
1800 \bool_lazy_all:nT
1801 {
1802     { \l__enumext_standar_bool }
1803     { \int_compare_p:nNn { \l__enumext_level_int } > { \c_zero_int } }
1804     { \bool_not_p:n { \g__enumext_starred_bool } }
1805     { \int_compare_p:nNn { \l__enumext_level_h_int } > { \c_zero_int } }
1806 }
1807 {
1808     \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
1809     {
1810         \int_step_function:nnN { 1 } { \l__enumext_level_int } \__enumext_tmp:n
1811         . \tl_use:N \l__enumext_label_copy_vii_tl
1812     }
1813 }

```

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

```

1814 \tl_put_right:Ne \l__enumext_newlabel_arg_one_tl
1815 {
1816     \l__enumext_store_name_tl \c_colon_str
1817     \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } }
1818 }

```

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.

```

1819 \tl_put_right:Ne \l__enumext_store_write_aux_file_tl
1820 {
1821     \__enumext_newlabel:nn
1822     { \exp_not:V \l__enumext_newlabel_arg_one_tl }
1823     { \l__enumext_newlabel_arg_two_tl }
1824 }
1825 \l__enumext_store_write_aux_file_tl
1826 }

```

(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.

```

1827 \cs_new_protected:Npn \__enumext_store_anskey_show_wrap:n #1
1828 {
1829     \par
1830     \bool_if:NT \l__enumext_starred_bool
1831     {
1832         \cs_set:Nn \__enumext_level: { vii }
1833     }
1834     \__enumext_print_keyans_box:cc
1835     { l__enumext_labelwidth_ \__enumext_level: _dim }
1836     { l__enumext_labelsep_ \__enumext_level: _dim }
1837     \__enumext_anskey_wrapper:n { #1 }
1838 }

```

(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.

```

1839 \cs_new_protected:Npn \__enumext_store_anskey_show_left:n #1
1840 {
1841     \bool_if:NT \l__enumext_show_answer_bool
1842     {
1843         \__enumext_store_anskey_show_wrap:n { #1 }
1844     }
1845     \bool_if:NT \l__enumext_show_position_bool

```



```

1846     {
1847         \tl_set:Nc \l__enumext_mark_answer_sym_tl
1848         {
1849             \group_begin:
1850             \exp_not:N \normalfont
1851             \exp_not:N \footnotesize [ \int_eval:n
1852             {
1853                 \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
1854             }
1855             ]
1856             \group_end:
1857         }
1858         \__enumext_store_anskey_show_wrap:n { #1 }
1859     }
1860 }

```

(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 current *⟨label⟩* \l__enumext_label_v_tl for the **keyans** environment and the current *⟨label⟩* \l__enumext_label_vi_tl for the **keyanspic** environment when using \item* and \anspic*, followed by the *contents* of the optional argument of both commands to the \l__enumext_store_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.

```

1861 \cs_new_protected:Npn \__enumext_keyans_addto_prop:n #1
1862 {
1863     \tl_clear:N \l__enumext_store_keyans_label_tl
1864     \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
1865     {
1866         \tl_put_right:Nc \l__enumext_store_keyans_label_tl { \l__enumext_label_vi_tl }
1867     }
1868     {
1869         \tl_put_right:Nc \l__enumext_store_keyans_label_tl { \l__enumext_label_v_tl }
1870     }
1871     \tl_if_novalue:nF { #1 }
1872     {
1873         \tl_put_right:Nc \l__enumext_store_keyans_label_tl { \c_space_tl #1 }
1874     }
1875     \__enumext_store_addto_prop:V \l__enumext_store_keyans_label_tl
1876 }

```

(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.

```

1877 \cs_new_protected:Nn \__enumext_keyans_store_ref:
1878 {
1879     \bool_if:NT \l__enumext_store_ref_key_bool
1880     {
1881         \cs_set_protected:Npn \__enumext_tmp:n #1
1882         {
1883             \tl_set_eq:cc { \l__enumext_label_copy_##1_tl } { \l__enumext_label_##1_tl }
1884             \tl_reverse:c { \l__enumext_label_copy_##1_tl }
1885             \tl_remove_once:cn { \l__enumext_label_copy_##1_tl } { . }
1886             \tl_reverse:c { \l__enumext_label_copy_##1_tl }
1887         }
1888         \clist_map_inline:nn { i, v, vi, vii, viii } { \__enumext_tmp:n {##1} }
1889         \__enumext_keyans_store_ref_aux_i:
1890     }
1891 }

```

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

```

1892 \cs_new_protected:Nn \__enumext_keyans_store_ref_aux_i:
1893 {
1894   \bool_if:NT \g__enumext_starred_bool
1895   {
1896     \tl_set_eq:NN \l__enumext_label_copy_i_tl \l__enumext_label_copy_vii_tl
1897   }
1898   \int_compare:nNnT { \l__enumext_keyans_pic_level_int } = { 1 }
1899   {
1900     \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
1901       { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_vi_tl }
1902   }
1903   \int_compare:nNnT { \l__enumext_keyans_level_int } = { 1 }
1904   {
1905     \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
1906       { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_v_tl }
1907   }
1908   \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
1909   {
1910     \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
1911       { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_viii_tl }
1912   }
1913   \tl_put_right:Ne \l__enumext_newlabel_arg_one_tl
1914     {
1915       \l__enumext_store_name_tl \c_colon_str
1916       \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } }
1917     }
1918   \__enumext_keyans_store_ref_aux_ii:
1919 }

```

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.

```

1920 \cs_new_protected:Nn \__enumext_keyans_store_ref_aux_ii:
1921 {
1922   \tl_put_right:Ne \l__enumext_store_write_aux_file_tl
1923     {
1924       \__enumext_newlabel:nn
1925         { \exp_not:V \l__enumext_newlabel_arg_one_tl }
1926         { \l__enumext_newlabel_arg_two_tl }
1927     }
1928   \l__enumext_store_write_aux_file_tl
1929 }

```

(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 $\langle label \rangle$ `\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 $\langle contents \rangle$ 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.

```

1930 \cs_new_protected:Npn \__enumext_keyans_addto_seq:n #1
1931 {
1932   \tl_clear:N \l__enumext_store_keyans_label_tl
1933   \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
1934   {
1935     \tl_put_right:Ne \l__enumext_store_keyans_label_tl { \item \l__enumext_label_vi_tl }
1936   }
1937   {
1938     \tl_put_right:Ne \l__enumext_store_keyans_label_tl { \item \l__enumext_label_v_tl }
1939   }
1940   \tl_if_novalue:nF { #1 }
1941   {
1942     \tl_put_right:Ne \l__enumext_store_keyans_label_tl { \c_space_tl #1 }
1943   }
1944   \__enumext_keyans_addto_seq_link:
1945 }

```

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.

```

1946 \cs_new_protected:Nn \__enumext_keyans_addto_seq_link:
1947 {
1948   \bool_lazy_and:nnT
1949     { \l__enumext_store_ref_key_bool }
1950     { \l__enumext_hyperref_bool }
1951   {
1952     \tl_put_right:Nx \l__enumext_store_keyans_label_tl
1953     {
1954       \hfill \exp_not:N \hyperlink
1955       {
1956         \exp_not:V \l__enumext_newlabel_arg_one_tl
1957       }
1958       { \exp_not:V \l__enumext_mark_ref_sym_tl }
1959     }
1960   }
1961   \__enumext_store_addto_seq:V \l__enumext_store_keyans_label_tl
1962   \tl_gset:NV \g__enumext_check_ans_item_tl \l__enumext_store_keyans_label_tl
1963   \bool_if:NT \l__enumext_check_ans_bool
1964   {
1965     \int_gincr:N \g__enumext_count_item_with_ans_int
1966   }
1967 }

```

(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.

```

1968 \cs_new_protected:Npn \__enumext_keyans_check_ans:nn #1 #2
1969 {
1970   \tl_if_empty:NTF \g__enumext_check_ans_item_tl
1971   {
1972     \msg_warning:nnnn { enumext } { missing-starred } { #1 } { #2 }
1973   }
1974   { \tl_gclear:N \g__enumext_check_ans_item_tl }
1975 }

```

(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* `\item*` and `\position` of stored content in `\prop list` for `keyans` and `keyanspic`. Need add 1 to `\l__enumext_show_position_bool` for `keyans` environment.

```

1976 \cs_new_protected:Npn \__enumext_keyans_show_left:n #1
1977 {
1978   \bool_if:NT \l__enumext_show_answer_bool
1979   {
1980     \tl_put_left:Nn \l__enumext_label_v_tl
1981     {
1982       \__enumext_print_keyans_box:NN
1983       \l__enumext_labelwidth_i_dim
1984       \l__enumext_labelsep_i_dim
1985     }
1986     \tl_if_novalue:nF { #1 }
1987     { \tl_put_right:Nn \l__enumext_label_v_tl { \c_space_tl [ #1 ] } }
1988   }
1989   \bool_if:NT \l__enumext_show_position_bool
1990   {
1991     \tl_set:Nx \l__enumext_mark_answer_sym_tl
1992     {
1993       \group_begin:

```

```

1994         \exp_not:N \normalfont
1995         \exp_not:N \footnotesize [ \int_eval:n
1996         {
1997             \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } + \c_one_int
1998         }
1999         ]
2000     \group_end:
2001 }
2002 \tl_put_left:Nn \l__enumext_label_v_tl
2003 {
2004     \__enumext_print_keyans_box:NN
2005     \l__enumext_labelwidth_i_dim
2006     \l__enumext_labelsep_i_dim
2007 }
2008 \tl_if_novalue:nF { #1 }
2009 { \tl_put_right:Nn \l__enumext_label_v_tl { \c_space_tl [ #1 ] } }
2010 }
2011 }

```

(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*
2012 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
2013 {
2014     \keys_define:nn { enumext / #1 }
2015     {
2016         item-sym* .tl_set:c = { \__enumext_item_symbol_#2_tl },
2017         item-sym* .value_required:n = true,
2018         item-sym* .initial:n = { $\star$ },
2019         item-pos* .dim_set:c = { \__enumext_item_symbol_sep_#2_dim },
2020         item-pos* .value_required:n = true,
2021     }
2022 }
2023 \clist_map_inline:nn
2024 {
2025     {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {enumext*}{vii}
2026 }
2027 { \__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](#).

```

2028 \cs_new_protected:Nn \__enumext_footnotetext:nn
2029 {
2030     \footnotetext[#1]{#2}
2031 }
2032 \cs_new_protected:Nn \__enumext_renew_footnote:
2033 {
2034     \seq_gclear:N \g__enumext_footnote_arg_seq
2035     \seq_gclear:N \g__enumext_footnote_int_seq
2036     \RenewDocumentCommand \footnote { o +m }
2037     {
2038         \tl_if_novalue:nTF {##1}
2039         {
2040             \stepcounter{footnote}
2041             \int_gset_eq:Nc \g__enumext_footnote_int { c@footnote }
2042         }
2043         {
2044             \int_gset:Nn \g__enumext_footnote_int { ##1 }
2045         }
2046         \footnotemark [ \g__enumext_footnote_int ]
2047         \seq_gput_right:Nn \g__enumext_footnote_arg_seq { ##2 }
2048         \seq_gput_right:NV \g__enumext_footnote_int_seq \g__enumext_footnote_int

```

```

2049     }
2050   }
2051   \cs_new_protected:Nn \__enumext_print_footnote:
2052   {
2053     \seq_if_empty:NF \g__enumext_footnote_int_seq
2054     {
2055       \seq_map_pairwise_function:NNN
2056       \g__enumext_footnote_int_seq
2057       \g__enumext_footnote_arg_seq
2058       \__enumext_footnotetext:nn
2059     }
2060   }

```

(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`

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

First we will see if the optional argument is present, if it is NOT present we will check the state of the variable `\l__enumext_check_ans_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).

```

2061 \cs_new_protected:Npn \__enumext_default_item:n #1
2062 {
2063   \tl_if_novalue:nTF {#1}
2064   {
2065     \bool_if:NT \l__enumext_check_ans_bool
2066     {
2067       \int_gincr:N \g__enumext_count_item_all_int
2068       \int_gincr:c { g__enumext_count_level_ \__enumext_level: _int }
2069     }
2070     \bool_set_true:c { l__enumext_wrap_label_ \__enumext_level: _bool }
2071     \__enumext_item_std:w \tl_use:c { l__enumext_fake_item_indent_ \__enumext_level: _tl }
2072   }
2073   {
2074     \bool_set_eq:cc
2075     { l__enumext_wrap_label_ \__enumext_level: _bool }
2076     { l__enumext_wrap_label_opt_ \__enumext_level: _bool }
2077     \__enumext_item_std:w [#1] \tl_use:c { l__enumext_fake_item_indent_ \__enumext_level: _tl }
2078   }
2079 }

```

(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).

```

2080 \cs_new_protected:Npn \__enumext_starred_item:nn #1 #2
2081 {
2082   \tl_if_novalue:nF {#1}
2083   {
2084     \tl_set:cn { l__enumext_item_symbol_ \__enumext_level: _tl } {#1}
2085   }
2086   \tl_gset_eq:Nc \g__enumext_item_symbol_tl { l__enumext_item_symbol_ \__enumext_level: _tl }
2087   \tl_if_novalue:nTF {#2}
2088   {
2089     \dim_set_eq:cc
2090     { l__enumext_item_symbol_sep_ \__enumext_level: _dim }
2091     { l__enumext_labelsep_ \__enumext_level: _dim }
2092   }
2093   {
2094     \dim_set:cn { l__enumext_item_symbol_sep_ \__enumext_level: _dim } {#2}
2095   }
2096   \bool_if:NT \l__enumext_check_ans_bool
2097   {
2098     \int_gincr:N \g__enumext_count_item_all_int
2099     \int_gincr:c { g__enumext_count_level_ \__enumext_level: _int }
2100   }
2101   \bool_set_true:c { l__enumext_wrap_label_ \__enumext_level: _bool }
2102   \__enumext_item_std:w \tl_use:c { l__enumext_fake_item_indent_ \__enumext_level: _tl }
2103 }

```

(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).

```

2104 \cs_new_protected:Nn \__enumext_redefine_item:
2105 {
2106   \RenewDocumentCommand \item { s o o }
2107   {
2108     \bool_if:nTF {##1}
2109     {
2110       \__enumext_starred_item:nn {##2} {##3}
2111     }
2112     { \__enumext_default_item:n {##2} }
2113   }
2114 }

```

(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`.

```

2115 \cs_new_protected:Npn \__enumext_keyans_default_item:n #1
2116 {
2117   \tl_if_novalue:nTF { #1 }
2118   {
2119     \bool_set_true:N \l__enumext_wrap_label_v_bool
2120     \__enumext_item_std:w \tl_use:N \l__enumext_fake_item_indent_v_tl
2121   }
2122   {
2123     \bool_set_eq:NN \l__enumext_wrap_label_v_bool \l__enumext_wrap_label_opt_v_bool
2124     \__enumext_item_std:w [#1] \tl_use:N \l__enumext_fake_item_indent_v_tl
2125   }
2126 }

```

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

`__enumext_keyans_starred_item:n`

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

```
2127 \cs_new_protected:Npn \__enumext_keyans_starred_item:n #1
2128 {
2129   \tl_set_eq:NN \l__enumext_keyans_tmpa_tl \l__enumext_label_v_tl
2130   \__enumext_keyans_show_left:n { #1 }
2131   \bool_set_true:N \l__enumext_wrap_label_v_bool
2132   \__enumext_item_std:w \tl_use:N \l__enumext_fake_item_indent_v_tl
```

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

```
2133   \tl_set_eq:NN \l__enumext_label_v_tl \l__enumext_keyans_tmpa_tl
2134   \__enumext_keyans_addto_prop:n { #1 }
2135   \__enumext_keyans_store_ref:
2136   \__enumext_keyans_addto_seq:n { #1 }
2137 }
```

(End of definition for `__enumext_keyans_starred_item:n`)

`\item*`
`__enumext_keyans_redefine_item:`

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

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

```
2138 \cs_new_protected:Nn \__enumext_keyans_redefine_item:
2139 {
2140   \RenewDocumentCommand \item { s o }
2141   {
2142     \bool_if:nTF {##1}
2143     {
2144       \peek_remove_spaces:n
2145       {
2146         \__enumext_keyans_starred_item:n {##2}
2147       }
2148     }
2149     {
2150       \__enumext_keyans_default_item:n {##2}
2151     }
2152   }
2153 }
```

(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.

```
2154 \cs_new_protected:Nn \__enumext_item_starred:
2155 {
2156   \tl_if_empty:cF { \l__enumext_item_symbol_ \l__enumext_level: _tl }
2157   {
2158     \mode_leave_vertical:
2159     \skip_horizontal:n { -\dim_use:c { \l__enumext_item_symbol_sep_ \l__enumext_level: _dim } }
2160     \makebox[ 0pt ][ r ]{ \tl_use:N \g__enumext_item_symbol_tl }
2161     \skip_horizontal:n { \dim_use:c { \l__enumext_item_symbol_sep_ \l__enumext_level: _dim } }
2162   }
2163 }
```

(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).

```

2164 \cs_new_protected:Nn \__enumext_make_label:
2165 {
2166   \RenewDocumentCommand \makelabel { m }
2167   {
2168     \tl_use:c { \__enumext_label_fill_left_ \__enumext_level: _tl }
2169     \tl_use:c { \__enumext_label_font_style_ \__enumext_level: _tl }
2170     \bool_if:cTF { \__enumext_wrap_label_ \__enumext_level: _bool }
2171     {
2172       \__enumext_item_starred:
2173       \use:c { \__enumext_wrapper_label_ \__enumext_level: :n } { ##1 }
2174     }
2175     { ##1 }
2176     \tl_use:c { \__enumext_label_fill_right_ \__enumext_level: _tl }
2177     \tl_gclear:N \g__enumext_item_symbol_tl
2178   }
2179 }

```

(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).

```

2180 \cs_new_protected:Nn \__enumext_keyans_make_label:
2181 {
2182   \RenewDocumentCommand \makelabel { m }
2183   {
2184     \tl_use:N \l__enumext_label_fill_left_v_tl
2185     \tl_use:N \l__enumext_label_font_style_v_tl
2186     \bool_if:NTF { \l__enumext_wrap_label_v_bool }
2187     {
2188       \__enumext_wrapper_label_v:n { ##1 }
2189     }
2190     { ##1 }
2191     \tl_use:N \l__enumext_label_fill_right_v_tl
2192   }
2193 }

```

(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.

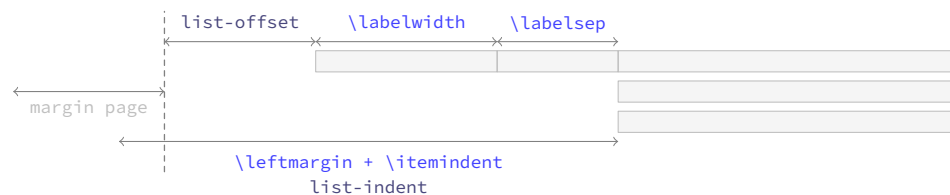


Figure 9: Representation of standard horizontal lengths in `list` environment.

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.

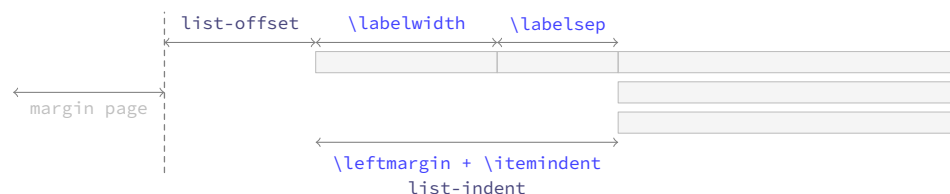


Figure 10: Representation of horizontal lengths concept in list in `enumext`.

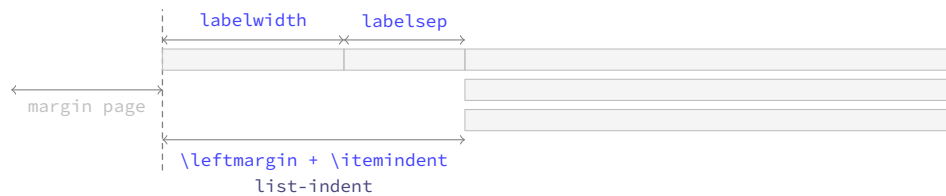
Where the default values will look like in the figure 11.

```

\__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:

Figure 11: Default horizontal lengths in `enumext`.

```

#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).

```

2194 \cs_new_protected:Npn \__enumext_calc_hspace:NNNNNNN #1 #2 #3 #4 #5 #6 #7
2195 {
2196   \dim_compare:nNnT { #1 } < { \c_zero_dim }
2197   {
2198     \msg_warning:nnnV { enumext } { width-non-positive } { labelwidth } { #1 }
2199     \dim_set:Nn #1 { \dim_abs:n { #1 } }
2200   }
2201   \dim_compare:nNnT { #2 } < { \c_zero_dim }
2202   {
2203     \msg_warning:nnnV { enumext } { width-negative } { labelsep } { #2 }
2204     \dim_set:Nn #2 { \dim_abs:n { #2 } }
2205   }

```

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

```

2206   \bool_if:nF #7 { \dim_set:Nn #4 { #1 + #2 } }

```

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

```

2207   \dim_compare:nNnTF { #4 } < { \c_zero_dim }
2208   {
2209     \dim_set:Nn #6 { #1 + #2 - #4 }
2210     \dim_set:Nn #5 { #1 + #2 + #3 - #6 }
2211   }
2212   {
2213     \dim_compare:nNnT { #4 } = { #1 + #2 }
2214     { \dim_set:Nn #6 { \c_zero_dim } }
2215     \dim_compare:nNnT { #4 } < { #1 + #2 }
2216     { \dim_set:Nn #6 { #1 + #2 - #4 } }
2217     \dim_compare:nNnT { #4 } > { #1 + #2 }
2218     {
2219       \dim_set:Nn #6 { -#1 - #2 + #4 }
2220       \dim_set:Nn #6 { #6*-1 }
2221     }
2222     \dim_set:Nn #5 { #1 + #2 + #3 - #6 }
2223   }
2224 }
2225 \cs_generate_variant:Nn \__enumext_calc_hspace:NNNNNNN { ccccccc }

```

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

10.31 Setting second argument of the lists

At this point of the code we have already programmed the necessary tools to create a custom `list` environment, remember that the function `__enumext_start_list:n` 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.

```

2226 \cs_set_protected:Npn \__enumext_tmp:n #1
2227 {

```

```

2228 \cs_new_protected:cpn { __enumext_list_arg_two_#1: }
2229 {
2230   \__enumext_calc_hspace:ccccc
2231   { \__enumext_labelwidth_#1_dim } { \__enumext_labelsep_#1_dim }
2232   { \__enumext_listoffset_#1_dim } { \__enumext_leftmargin_tmp_#1_dim }
2233   { \__enumext_leftmargin_#1_dim } { \__enumext_itemindent_#1_dim }
2234   { \__enumext_leftmargin_tmp_#1_bool }
2235 \clist_map_inline:nn
2236 { labelsep, labelwidth, itemindent, leftmargin, rightmargin, listparindent }
2237 { \dim_set_eq:cc {####1} { \__enumext_####1_#1_dim } }
2238 \clist_map_inline:nn { topsep, parsep, partopsep, itemsep }
2239 { \skip_set_eq:cc {####1} { \__enumext_####1_#1_skip } }
2240 \usecounter { enumX#1 }
2241 \bool_lazy_and:nnTF
2242 { \str_if_eq_p:nn {#1} { i } }
2243 { \bool_if_p:N \__enumext_resume_bool }
2244 { \setcounter { enumXi } { \int_eval:n { \g__enumext_resume_int } } }
2245 {
2246   \setcounter { enumX#1 }
2247   { \int_eval:n { \int_use:c { \__enumext_start_#1_int } - 1 } }
2248 }
2249 \str_if_eq:nnTF {#1} { v }
2250 {
2251   \__enumext_keyans_redefine_item:
2252   \__enumext_keyans_make_label:
2253   \__enumext_keyans_fake_item:
2254   \bool_if:cT { \__enumext_show_length_#1_bool }
2255   {
2256     \msg_term:nnnn { enumext } { list-lengths-not-nested } { v } { keyans }
2257   }
2258 }
2259 {
2260   \__enumext_redefine_item:
2261   \__enumext_make_label:
2262   \__enumext_use_key_ref:
2263   \__enumext_fake_item:
2264   \bool_if:cT { \__enumext_show_length_#1_bool }
2265   {
2266     \msg_term:nnne { enumext } { list-lengths } {#1} { \int_use:N \__enumext_level_int }
2267   }
2268 }
2269 }
2270 }
2271 \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.

```

2272 \cs_set_protected:Npn \__enumext_tmp:n #1
2273 {
2274   \cs_new_protected:cpn { __enumext_list_arg_two_#1: }
2275   {
2276     \__enumext_calc_hspace:ccccc
2277     { \__enumext_labelwidth_#1_dim } { \__enumext_labelsep_#1_dim }
2278     { \__enumext_listoffset_#1_dim } { \__enumext_leftmargin_tmp_#1_dim }
2279     { \__enumext_leftmargin_#1_dim } { \__enumext_itemindent_#1_dim }
2280     { \__enumext_leftmargin_tmp_#1_bool }
2281 \clist_map_inline:nn
2282 { labelsep, labelwidth, itemindent, leftmargin, rightmargin, listparindent }
2283 { \dim_set_eq:cc {####1} { \__enumext_####1_#1_dim } }
2284 \clist_map_inline:nn { topsep, parsep, partopsep, itemsep }
2285 { \skip_set_eq:cc {####1} { \__enumext_####1_#1_skip } }
2286 \skip_set_eq:Nc \parsep { \__enumext_itemsep_#1_skip }
2287 \skip_zero:N \partopsep
2288 \usecounter { enumX#1 }
2289 \bool_lazy_and:nnTF
2290 { \str_if_eq_p:nn {#1} { vii } } { \bool_if_p:N \__enumext_resume_vii_bool }
2291 { \setcounter { enumXvii } { \int_eval:n { \g__enumext_resume_vii_int } } }

```

```

2292     {
2293       \setcounter { enumX#1 }
2294       { \int_eval:n { \int_use:c { l__enumext_start_#1_int } - 1 } }
2295     }
2296     \__enumext_use_key_ref_h:
2297     \str_if_eq:nnTF {#1} { vii }
2298     {
2299       \__enumext_fake_item_vii:
2300       \bool_if:cT { l__enumext_show_length_vii_bool }
2301       { \msg_term:nnnn { enumext } { list-lengths-not-nested } { vii } { enumext* } }
2302     }
2303     {
2304       \__enumext_fake_item_viii:
2305       \bool_if:cT { l__enumext_show_length_#1_bool }
2306       { \msg_term:nnnn { enumext } { list-lengths-not-nested } { #1 } { keyans* } }
2307     }
2308   }
2309 }
2310 \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.

```

2311 \NewDocumentEnvironment{enumext}{ 0{ } }
2312 {
2313   \__enumext_safe_exec:
2314   \__enumext_parse_keys:n {#1}
2315   \__enumext_before_list:
2316   \__enumext_start_store_level:
2317   \__enumext_start_list:nn
2318   { \tl_use:c { l__enumext_label_ \__enumext_level: _tl } }
2319   {
2320     \use:c { __enumext_list_arg_two_ \__enumext_level: : }
2321     \__enumext_before_keys_exec:
2322   }
2323   \__enumext_after_args_exec:
2324 }
2325 {
2326   \__enumext_stop_list:
2327   \__enumext_stop_store_level:
2328   \__enumext_after_list:
2329 }

```

(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.

```

2330 \cs_new_protected:Nn \__enumext_safe_exec:
2331 {
2332   \int_incr:N \l__enumext_level_int
2333   \int_compare:nNnT { \l__enumext_level_int } > { 4 }
2334   { \msg_fatal:nn { enumext } { list-too-deep } }
2335   \bool_set_true:N \l__enumext_standar_bool
2336   \bool_lazy_all:nT
2337   {
2338     { \bool_not_p:n { \l__enumext_starred_bool } }
2339     { \int_compare_p:nNn { \l__enumext_level_h_int } = { \c_zero_int } }
2340   }
2341   {
2342     \bool_gset_true:N \g__enumext_standar_bool
2343   }
2344 }

```

(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.

```

2345 \cs_new_protected:Npn \__enumext_parse_keys:n #1
2346 {
2347   \exp_args:Ne \keys_set:nn
2348   { enumext / level-\int_use:N \__enumext_level_int } {#1}
2349   \bool_if:NT \l__enumext_store_active_bool
2350   {
2351     \__enumext_parse_store_keys:n {#1}
2352   }
2353 }

```

(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:`.

```

2354 \cs_new_protected:Npn \__enumext_parse_store_keys:n #1
2355 {
2356   \bool_if:cF { \l__enumext_store_columns_ \__enumext_level: _bool }
2357   {
2358     \regex_match:nnT { \b columns\b } {#1}
2359     {
2360       \int_set_eq:cc
2361       { \l__enumext_store_columns_ \__enumext_level: _int }
2362       { \l__enumext_columns_ \__enumext_level: _int }
2363       \tl_put_right:ce { \l__enumext_store_opt_ \__enumext_level: _tl }
2364       {
2365         columns = \exp_not:v { \l__enumext_store_columns_ \__enumext_level: _int },
2366       }
2367     }
2368   }
2369   \bool_if:cF { \l__enumext_store_columns_sep_ \__enumext_level: _bool }
2370   {
2371     \regex_match:nnT { \b columns-sep\b } {#1}
2372     {
2373       \dim_set_eq:cc
2374       { \l__enumext_store_columns_sep_ \__enumext_level: _dim }
2375       { \l__enumext_columns_sep_ \__enumext_level: _dim }
2376       \tl_put_right:ce { \l__enumext_store_opt_ \__enumext_level: _tl }
2377       {
2378         columns-sep = \exp_not:v { \l__enumext_store_columns_sep_ \__enumext_level: _dim },
2379       }
2380     }
2381   }
2382 }

```

(End of definition for __enumext_parse_store_keys:n)

__enumext_start_store_level:

__enumext_stop_store_level:

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

```

2383 \cs_new_protected:Nn \__enumext_start_store_level:
2384 {
2385   %\bool_lazy_and:nnT
2386   %{ \l__enumext_store_active_bool }
2387   %{ \bool_not_p:n { \l__enumext_keyans_env_bool } }
2388   %{
2389     %\int_compare:nNnT { \l__enumext_level_h_int } = { 1 }
2390     %{
2391       %\bool_set_true:c { \l__enumext_store_upper_level_ \__enumext_level: _bool }
2392       %\__enumext_store_level_open:
2393     }
2394     %\int_compare:nNnT { \l__enumext_level_int } > { 1 }
2395     %{
2396       \bool_set_true:c { \l__enumext_store_upper_level_ \__enumext_level: _bool }
2397       \__enumext_store_level_open:
2398     }
2399   }%}
2400 }
2401 \cs_new_protected:Nn \__enumext_stop_store_level:

```

```

2402 {
2403   \bool_if:cT { \__enumext_store_upper_level_ \__enumext_level: _bool }
2404   {
2405     \__enumext_store_level_close:
2406   }
2407 }

```

(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.

```

2408 \cs_new_protected:Nn \__enumext_before_list:
2409 {
2410   \__enumext_vspace_above:
2411   \__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.

```

2412   \__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.

```

2413   \dim_compare:nNt
2414   { \dim_use:c { \l__enumext_minipage_right_ \__enumext_level: _dim } } > { \c_zero_dim }
2415   {
2416     \dim_set:cn { \l__enumext_minipage_left_ \__enumext_level: _dim }
2417     {
2418       \linewidth
2419       - \dim_use:c { \l__enumext_minipage_right_ \__enumext_level: _dim }
2420       - \dim_use:c { \l__enumext_minipage_hsep_ \__enumext_level: _dim }
2421     }

```

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

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

```

2422   \bool_set_true:c { \l__enumext_minipage_active_ \__enumext_level: _bool }
2423   \int_gincr:N \g__enumext_minipage_stat_int
2424   \__enumext_mini_addvspace:
2425   \nointerlineskip\noindent
2426   \begin{\__enumext_mini_env*}
2427   { \dim_use:c { \l__enumext_minipage_left_ \__enumext_level: _dim } }
2428   }
2429   \__enumext_multicols_start:
2430   }

```

(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.

```

2431 \cs_new_protected:Nn \__enumext_multicols_start:
2432 {
2433   \int_compare:nNt
2434   { \int_use:c { \l__enumext_columns_ \__enumext_level: _int } } > { 1 }
2435   {
2436     \dim_compare:nNt
2437     { \dim_use:c { \l__enumext_columns_sep_ \__enumext_level: _dim } } = { \c_zero_dim }
2438     {
2439       \dim_set:cn { \l__enumext_columns_sep_ \__enumext_level: _dim }
2440       {
2441         ( \dim_use:c { \l__enumext_labelwidth_ \__enumext_level: _dim }
2442         + \dim_use:c { \l__enumext_labelsep_ \__enumext_level: _dim }

```

```

2443         ) / \int_use:c { l__enumext_columns_ \__enumext_level: _int }
2444         - \dim_use:c { l__enumext_listoffset_ \__enumext_level: _dim }
2445     }
2446 }
2447 \dim_set_eq:Nc \columnsep { l__enumext_columns_sep_ \__enumext_level: _dim }
2448 \skip_zero:N \multicolsep
2449 \int_compare:nNnT { \l__enumext_level_int } > { 1 }
2450 {
2451     \dim_zero:N \columnseprule
2452 }

```

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.

```

2453     \bool_if:cF { l__enumext_minipage_active_ \__enumext_level: _bool }
2454     {
2455         \__enumext_multi_addvspace:
2456     }
2457     \raggedcolumns
2458     \begin{multicols}{ \int_use:c { l__enumext_columns_ \__enumext_level: _int } }
2459 }
2460 }

```

(End of definition for `__enumext_multicols_start:`)

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

```

2461 \cs_new_protected:Nn \__enumext_multicols_stop:
2462 {
2463     \int_compare:nNnT
2464     { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } > { 1 }
2465     {
2466         \end{multicols}
2467         \bool_if:cF { l__enumext_minipage_active_ \__enumext_level: _bool }
2468         {
2469             \par\addvspace{ \skip_use:c { l__enumext_multicols_below_ \__enumext_level: _skip } }
2470         }
2471     }

```

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.

```

2472     \bool_lazy_and:nnT
2473     { \l__enumext_check_ans_bool }
2474     { \bool_not_p:n { \g__enumext_starred_bool } }
2475     {
2476         \bool_gset_true:N \g__enumext_check_ans_show_bool
2477         \tl_gset:NV \g__enumext_store_name_tl \l__enumext_store_name_tl
2478     }
2479 }

```

(End of definition for `__enumext_multicols_stop:`)

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

```

2480 \cs_new_protected:Nn \__enumext_after_list:
2481 {
2482     \bool_if:cTF { l__enumext_minipage_active_ \__enumext_level: _bool }
2483     {
2484         \int_compare:nNnT { \g__enumext_minipage_stat_int } = { 1 }
2485         {
2486             \msg_warning:nn { enumext } { missing-miniright }
2487             \miniright
2488         }
2489         \int_gzero:N \g__enumext_minipage_stat_int
2490         \end{__enumext_mini_env*}

```



```

2491     \par\addvspace { \l__enumext_minipage_after_skip }
2492   }
2493   { \__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.

```

2494   \__enumext_after_stop_list:
2495   \__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.

```

2496   \bool_set_false:N \l__enumext_standar_bool
2497   \int_gset_eq:NN \g__enumext_resume_int \value{enumXi}
2498   \int_if_exist:cT { g__enumext_resume_ \l__enumext_store_name_tl _int }
2499   {
2500     \int_gset_eq:cN
2501     { g__enumext_resume_ \l__enumext_store_name_tl _int }
2502     { \value{enumXi} }
2503   }
2504 }

```

(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`.

```

2505 \__enumext_after_env:nn {enumext}
2506 {
2507   \bool_if:NT \g__enumext_check_ans_show_bool
2508   {
2509     \int_compare:nNnT { \l__enumext_level_int } = { 0 }
2510     {
2511       \__enumext_check_ans_active:
2512     }
2513   }
2514   \bool_gset_false:N \g__enumext_check_ans_show_bool
2515   \tl_gclear:N \g__enumext_store_name_tl
2516 }

```

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.

```

2517 \NewDocumentEnvironment{keyans}{ 0{} }
2518 {
2519   \__enumext_keyans_safe_exec:
2520   \__enumext_keyans_parse_keys:n {#1}
2521   \__enumext_before_list_v:
2522   \__enumext_start_list:nn
2523   { \tl_use:N \l__enumext_label_v_tl }
2524   {
2525     \__enumext_list_arg_two_v:
2526     \__enumext_before_keys_exec_v:
2527   }
2528   \__enumext_after_args_exec_v:
2529 }
2530 {
2531   \__enumext_keyans_check_ans:nn { item }{ keyans }
2532   \__enumext_stop_list:
2533   \__enumext_after_list_v:
2534 }

```

(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.

```

2535 \cs_new_protected:Nn \__enumext_keyans_safe_exec:
2536 {

```

```

2537 \bool_if:NF \l__enumext_store_active_bool
2538 {
2539   \msg_error:nnnn { enumext } { wrong-place }{ keyans }{ save-ans }
2540 }
2541 \int_incr:N \l__enumext_keyans_level_int
2542 \bool_set_true:N \l__enumext_keyans_env_bool
2543 % Set false for interfering with enumext nested in keyans (yes, its possible and crayze)
2544 \bool_set_false:N \l__enumext_store_active_bool
2545 \int_compare:nNnT { \l__enumext_keyans_level_int } > { 1 }
2546 {
2547   \msg_error:nn { enumext } { keyans-nested }
2548 }
2549 \int_compare:nNnT { \l__enumext_level_int } > { 1 }
2550 {
2551   \msg_error:nn { enumext } { keyans-wrong-level }
2552 }
2553 }

```

(End of definition for \l__enumext_keyans_safe_exec:.)

```

\__enumext_keyans_parse_keys:n Parse [key = val] for keyans environment.
2554 \cs_new_protected:Npn \__enumext_keyans_parse_keys:n #1
2555 {
2556   \keys_set:nn { enumext / keyans } {#1}
2557 }

```

(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.

```

2558 \cs_new_protected:Nn \__enumext_before_list_v:
2559 {
2560   \__enumext_vspace_above_v:
2561   \__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.

```

2562 \dim_compare:nNnT { \l__enumext_minipage_right_v_dim } > { \c_zero_dim }
2563 {
2564   \dim_set:Nn \l__enumext_minipage_left_v_dim
2565   {
2566     \linewidth - \l__enumext_minipage_right_v_dim - \l__enumext_minipage_hsep_v_dim
2567   }

```

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

```

2568 \bool_set_true:N \l__enumext_minipage_active_v_bool
2569 \int_gincr:N \g__enumext_minipage_stat_int
2570 \__enumext_keyans_mini_addvspace:
2571 \nointerlineskip\noindent
2572 \begin{__enumext_mini_env*}{ \l__enumext_minipage_left_v_dim }
2573 }

```

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

```

2574 \__enumext_keyans_multicols_start:
2575 }

```

(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.

```

2576 \cs_new_protected:Nn \__enumext_keyans_multicols_start:
2577 {
2578   \int_compare:nNnT { \l__enumext_columns_v_int } > { 1 }
2579   {

```

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

```

2580     \dim_compare:nNt { \l__enumext_columns_sep_v_dim } = { \c_zero_dim }
2581     {
2582         \dim_set:Nn \l__enumext_columns_sep_v_dim
2583         {
2584             (
2585                 \l__enumext_labelwidth_v_dim + \l__enumext_labelsep_v_dim
2586             ) / \l__enumext_columns_v_int
2587             - \l__enumext_listoffset_v_dim
2588         }
2589     }
2590     \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).

```

2591     \skip_zero:N \multicolsep
2592     \dim_zero:N \columnseprule

```

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

```

2593     \bool_if:NF \l__enumext_minipage_active_v_bool
2594     {
2595         \__enumext_keyans_multi_addvspace:
2596     }
2597     \raggedcolumns
2598     \begin{multicols}{ \l__enumext_columns_v_int }
2599 }
2600 }

```

(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.

```

2601 \cs_new_protected:Nn \__enumext_keyans_multicols_stop:
2602 {
2603     \int_compare:nNt { \l__enumext_columns_v_int } > { 1 }
2604     {
2605         \end{multicols}
2606         \bool_if:NF \l__enumext_minipage_active_v_bool
2607         {
2608             \par\addvspace{ \l__enumext_multicols_below_v_skip }
2609         }
2610     }
2611 }

```

(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.

```

2612 \cs_new_protected:Nn \__enumext_after_list_v:
2613 {
2614     \bool_if:NTF \l__enumext_minipage_active_v_bool
2615     {
2616         \int_compare:nNt { \g__enumext_minipage_stat_int } = { 1 }
2617         {
2618             \msg_warning:nn { enumext } { missing-miniright }
2619             \miniright
2620         }
2621         \int_gzero:N \g__enumext_minipage_stat_int
2622         \end{\__enumext_mini-env*}
2623         \par\addvspace{ \l__enumext_minipage_after_skip }
2624     }
2625     { \__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.

```
2626 \bool_set_false:N \__enumext_keyans_env_bool
2627 \__enumext_after_stop_list_v:
2628 \__enumext_vspace_below_v:
2629 }
```

(End of definition for `__enumext_after_list_v:`)

10.34 The environment `keyanspic` and `\anspic`

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

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

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

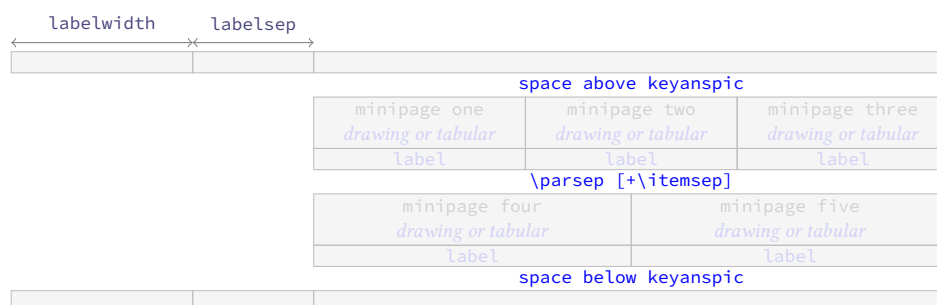


Figure 12: Representation of the `keyanspic` spacing in `enumext`.

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*[\content]` store the current `\label` next to the `[\content]` if it is present in the `\sequence` and `\prop list` defined by `save-ans` key. This command is used as a replacement for `\item` in the `keyanspic` environment.

```
2630 \NewDocumentCommand \anspic { s o +m }
2631 {
```

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

```
2632 \bool_if:NF \__enumext_store_active_bool
2633 {
2634 \msg_error:nnnn { enumext } { wrong-place } { keyanspic } { save-ans }
2635 }
2636 \int_compare:nNt { \__enumext_level_int } > { 1 }
2637 {
2638 \msg_error:nn { enumext } { keyanspic-wrong-level }
2639 }
2640 \int_compare:nNt { \__enumext_keyans_level_int } = { 1 }
2641 {
2642 \msg_error:nnnn { enumext } { command-wrong-place } { anspic } { keyans }
2643 }
```

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.

```
2644 \seq_put_right:Nn \__enumext_keyans_pic_body_seq
2645 {
2646 \__enumext_keyans_anspic_code:nnn { #1 } { #2 } { #3 }
2647 }
2648 }
```

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

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

```
2649 \cs_new_protected:Nn \__enumext_keyans_anspic_code:nnn
2650 {
2651 \stepcounter { enumXvi }
```

```

2652 #3 \
2653 \bool_if:nT { #1 }
2654 {
2655   \__enumext_keyans_addto_prop:n { #2 }
2656   \__enumext_keyans_store_ref:
2657   \__enumext_keyans_addto_seq:n { #2 }
2658   \bool_lazy_or:nnT
2659   { \__enumext_show_answer_bool }
2660   { \__enumext_show_position_bool }
2661   {
2662     \tl_set_eq:NN \__enumext_label_v_tl \__enumext_label_vi_tl
2663     \__enumext_keyans_show_left:n { #2 }
2664     \tl_set_eq:NN \__enumext_label_vi_tl \__enumext_label_v_tl
2665   }
2666 }
2667 \tl_use:N \__enumext_label_font_style_v_tl
2668 \__enumext_wrapper_label_v:n { \__enumext_label_vi_tl }
2669 }

```

(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.

```

2670 \NewDocumentEnvironment{keyanspic}{ o }
2671 {
2672   \__enumext_keyans_pic_safe_exec:
2673   \__enumext_start_list:nn
2674   { }
2675   {
2676     \__enumext_keyans_pic_arg_two:
2677   }

```

We apply the “adjusted” vertical spacing above the environment

```

2678   \vspace { \__enumext_keyans_pic_above_skip }
2679 }

```

If the optional argument is not present, the number of times the `\anspic` command appears will be counted from `__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.

```

2680 {
2681   \tl_if_novalue:nTF { #1 }
2682   {
2683     \__enumext_keyans_pic_do:e { \seq_count:N \__enumext_keyans_pic_body_seq }
2684   }
2685   { \__enumext_keyans_pic_do:n { #1 } }
2686   \__enumext_stop_list:
2687   \__enumext_keyans_check_ans:nn { anspic } { keyanspic }
2688   \setcounter { enumXvi } { 0 }
2689   \vspace { \__enumext_topsep_v_skip }
2690   %\bool_set_false:N \__enumext_store_active_bool
2691 }

```

(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.

```

2692 \cs_new_protected:Nn \__enumext_keyans_pic_safe_exec:
2693 {
2694   \int_incr:N \__enumext_keyans_pic_level_int
2695   \int_compare:nNnT { \__enumext_keyans_pic_level_int } > { 1 }
2696   {
2697     \msg_error:nn { enumext } { keyanspic-nested }
2698   }
2699 }

```

(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`.

```
2700 \cs_new_protected:Npn \__enumext_keyans_pic_skip_abs:N #1
2701 {
2702   \dim_compare:nNtT { #1 } < { 0pt }
2703   { \skip_set:Nn #1 { -#1 } }
2704 }
```

(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.

```
2705 \cs_new_protected:Nn \__enumext_keyans_pic_arg_two:
2706 {
```

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.

```
2707   \bool_set_false:N \l__enumext_leftmargin_tmp_v_bool
2708   \__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.

```
2709   \skip_add:Nn \parsep { \itemsep }
2710   \dim_add:Nn \leftmargin { -\labelwidth - \labelsep }
2711   \dim_zero:N \labelwidth
2712   \dim_zero:N \listparindent
2713   \dim_zero:N \labelsep
2714   \skip_zero:N \partopsep
2715   \skip_zero:N \itemsep
```

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

```
2716   \__enumext_keyans_pic_skip_abs:N \parsep
2717   \skip_set:Nn \l__enumext_keyans_pic_above_skip
2718   {
2719     \box_dp:N \strutbox
2720     + \l__enumext_topsep_v_skip
2721     - \parsep
2722   }
2723   \__enumext_item_std:w \scan_stop:
2724 }
```

(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`.

```
2725 \cs_new_protected:Nn \__enumext_keyans_pic_do:n
2726 {
2727   \clist_map_function:nN { #1 } \__enumext_keyans_pic_row:n
2728 }
2729 \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.

```
2730 \cs_new_protected:Nn \__enumext_keyans_pic_row:n
2731 {
2732   \dim_set:Nn \l__enumext_keyans_pic_width_dim { \linewidth / #1 }
2733   \int_set:Nn \l__enumext_keyans_pic_above_int { \l__enumext_keyans_pic_below_int }
2734   \int_set:Nn \l__enumext_keyans_pic_below_int { \l__enumext_keyans_pic_above_int + #1 }
2735   \int_step_inline:nnn
2736   { \l__enumext_keyans_pic_above_int + 1 }
2737   { \l__enumext_keyans_pic_below_int }
2738   {
2739     \__enumext_minipage:w [ b ]{ \l__enumext_keyans_pic_width_dim }
2740     \centering
```

```

2741         \seq_item:Nn \l__enumext_keyans_pic_body_seq { ##1 }
2742     \__enumext_endminipage:
2743 }
2744 \par
2745 }

```

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

10.35 The `enumext*` and `keyans*` environments

Generating horizontal list environments is NOT as simple as standard \TeX list environments. The fundamental part of the code is adapted from the `shortlst` package to a more modern version using `expl3`. It is not possible to redefine `\item` and `\makelabel` as in the non starred versions (at least I have not achieved it) and as we will make it behave differently, we have no other option than to define a cascade of functions.

To achieve the horizontal list environment we will capture the `\item` command and the content of this in an plain `lrbox` box using `\makebox` for the `label` and a `minipage` environment for the content passed to `\item`, we will also add the optional argument (`\langle number \rangle`) to `\item` to be able to *join columns* horizontally, in simple terms, we want `\item` to behave in the same way as in the `enumext` environment but adding an optional first argument (`\langle number \rangle`).

10.35.1 Functions for item box width

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

```

2746 \cs_new_protected:Nn \__enumext_starred_columns_set_vii:
2747 {
2748     \dim_compare:nNt { \l__enumext_columns_sep_vii_dim } = { \c_zero_dim }
2749     {
2750         \dim_set:Nn \l__enumext_columns_sep_vii_dim
2751         {
2752             ( \l__enumext_labelwidth_vii_dim + \l__enumext_labelsep_vii_dim )
2753             / \l__enumext_columns_vii_int
2754         }
2755     }
2756     \int_set:Nn \l__enumext_tmpa_vii_int { \l__enumext_columns_vii_int - \c_one_int }
2757     \dim_set:Nn \l__enumext_item_width_vii_dim
2758     {
2759         ( \linewidth - \l__enumext_columns_sep_vii_dim * \l__enumext_tmpa_vii_int )
2760         / \l__enumext_columns_vii_int - \l__enumext_labelwidth_vii_dim
2761         - \l__enumext_labelsep_vii_dim
2762     }
2763     \dim_zero_new:N \itemwidth
2764 }

```

(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`.

```

2765 \cs_new_protected:Npn \__enumext_starred_joined_item_vii:n #1
2766 {
2767     \int_set:Nn \l__enumext_joined_item_vii_int {#1}
2768     \int_compare:nNt { \l__enumext_joined_item_vii_int } > { \l__enumext_columns_vii_int }
2769     {
2770         \msg_warning:nnee { enumext } { item-joined }
2771         { \int_use:N \l__enumext_joined_item_vii_int }
2772         { \int_use:N \l__enumext_columns_vii_int }
2773         \int_set:Nn \l__enumext_joined_item_vii_int
2774         {
2775             \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + \c_one_int
2776         }
2777     }
2778     \int_compare:nNt
2779     { \l__enumext_joined_item_vii_int }
2780     >
2781     { \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + \c_one_int }
2782     {
2783         \msg_warning:nnee { enumext } { item-joined-columns }
2784         { \int_use:N \l__enumext_joined_item_vii_int }
2785         {
2786             \int_eval:n
2787             { \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + \c_one_int }

```



```

2788     }
2789     \int_set:Nn \l__enumext_joined_item_vii_int
2790     {
2791         \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + \c_one_int
2792     }
2793 }

```

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

```

2794 \int_compare:nNnTF { \l__enumext_joined_item_vii_int } > { \c_one_int }
2795 {
2796     \int_set_eq:NN \l__enumext_joined_item_aux_vii_int \l__enumext_joined_item_vii_int
2797     \int_decr:N \l__enumext_joined_item_aux_vii_int
2798     \int_add:Nn \l__enumext_item_column_pos_vii_int { \l__enumext_joined_item_aux_vii_int }
2799     \int_gadd:Nn \g__enumext_item_count_all_vii_int { \l__enumext_joined_item_aux_vii_int }
2800     \dim_set:Nn \l__enumext_joined_width_vii_dim
2801     {
2802         \l__enumext_item_width_vii_dim * \l__enumext_joined_item_vii_int
2803         + ( \l__enumext_labelwidth_vii_dim + \l__enumext_labelsep_vii_dim
2804             + \l__enumext_columns_sep_vii_dim
2805             ) * \l__enumext_joined_item_aux_vii_int
2806     }
2807     \dim_set_eq:NN \itemwidth \l__enumext_joined_width_vii_dim
2808 }
2809 {
2810     \dim_set_eq:NN \l__enumext_joined_width_vii_dim \l__enumext_item_width_vii_dim
2811     \dim_set_eq:NN \itemwidth \l__enumext_item_width_vii_dim
2812 }
2813 }

```

(End of definition for __enumext_starred_joined_item_vii:n.)

__enumext_start_mini_vii:

The implementation of the `mini-env` key support is almost identical to the one used in the `enumext` and `keyans` environments, the difference is that the `__enumext_mini_env*` environment on the “right side” is executed “after” closing the environment, so it is necessary to make a global copy of the variable `\l__enumext_minipage_right_vii_dim` in the variable `\g__enumext_minipage_right_vii_dim`.

```

2814 \cs_new_protected:Nn \__enumext_start_mini_vii:
2815 {
2816     \dim_compare:nNnT { \l__enumext_minipage_right_vii_dim } > { \c_zero_dim }
2817     {
2818         \dim_set:Nn \l__enumext_minipage_left_vii_dim
2819         {
2820             \linewidth
2821             - \l__enumext_minipage_right_vii_dim
2822             - \l__enumext_minipage_hsep_vii_dim
2823         }
2824         \bool_set_true:N \l__enumext_minipage_active_vii_bool
2825         \dim_gset_eq:NN
2826             \g__enumext_minipage_right_vii_dim
2827             \l__enumext_minipage_right_vii_dim
2828         \__enumext_mini_addvspace_vii:
2829         \nointerlineskip\noindent
2830         \begin{__enumext_mini_env*}{ \l__enumext_minipage_left_vii_dim }
2831     }
2832 }

```

(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”.

```

2833 \cs_new_protected:Nn \__enumext_stop_mini_vii:
2834 {
2835     \bool_if:NT \l__enumext_minipage_active_vii_bool
2836     {
2837         \end{__enumext_mini_env*}
2838         \hfill
2839         \bool_gset_true:N \g__enumext_minipage_active_vii_bool
2840     }
2841 }

```

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”.

```

2842 \__enumext_after_env:nn {enumext*}
2843 {
2844   \bool_if:NT \g__enumext_minipage_active_vii_bool
2845   {
2846     \begin{__enumext_mini_env*}{ \g__enumext_minipage_right_vii_dim }
2847     \par\addvspace { \g__enumext_minipage_right_skip }
2848     \bool_if:NF \g__enumext_minipage_center_vii_bool
2849     {
2850       \centering
2851     }
2852     \tl_use:N \g__enumext_miniright_code_vii_tl % the code
2853     \end{__enumext_mini_env*}
2854     \par\addvspace{ \g__enumext_minipage_after_skip }
2855   }
2856   \bool_gset_false:N \g__enumext_minipage_active_vii_bool
2857   \bool_gset_true:N \g__enumext_minipage_center_vii_bool
2858   \tl_gclear:N \g__enumext_miniright_code_vii_tl
2859   \dim_gzero:N \g__enumext_minipage_right_vii_dim
2860 }

```

(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.

```

2861 \NewDocumentEnvironment{enumext*}{ o }
2862 {
2863   \__enumext_safe_exec_vii:
2864   \__enumext_parse_keys_vii:n {#1}
2865   \__enumext_before_list_vii:
2866   \__enumext_start_store_level_vii:
2867   \__enumext_start_list:nn { }
2868   {
2869     \__enumext_list_arg_two_vii:
2870     \__enumext_before_keys_exec_vii:
2871   }
2872   \__enumext_starred_columns_set_vii:
2873   \item[] \scan_stop:
2874   \cs_set_eq:NN \__enumext_stop_item_tmp_vii: \noindent
2875   \cs_set_eq:NN \item \__enumext_start_item_tmp_vii:
2876 }
2877 {
2878   \__enumext_stop_item_tmp_vii:
2879   \__enumext_remove_extra_parsep_vii:
2880   \__enumext_stop_list:
2881   \__enumext_stop_store_level_vii:
2882   \__enumext_after_list_vii:
2883 }

```

(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`.

```

2884 \cs_new_protected:Nn \__enumext_safe_exec_vii:
2885 {
2886   \int_incr:N \l__enumext_level_h_int
2887   \int_compare:nNnT { \l__enumext_level_h_int } > { 1 }
2888   {
2889     \msg_error:nn { enumext } { nested }
2890   }
2891   \bool_set_true:N \l__enumext_starred_bool
2892   \bool_lazy_all:nT
2893   {
2894     { \bool_not_p:n { \l__enumext_standar_bool } }
2895     { \int_compare_p:nNn { \l__enumext_level_int } = { \c_zero_int } }
2896   }
2897   {
2898     \bool_gset_true:N \g__enumext_starred_bool
2899   }

```

```
2900 }
```

(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.

```
2901 \cs_new_protected:Npn \__enumext_parse_keys_vii:n #1
2902 {
2903   \tl_if_novalue:NF {#1}
2904   {
2905     \keys_set:nn { enumext / enumext* } {#1}
2906     \bool_if:NT \__enumext_store_active_bool
2907     {
2908       \__enumext_parse_store_keys_vii:n {#1}
2909     }
2910   }
2911 }
```

(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:`.

```
2912 \cs_new_protected:Npn \__enumext_parse_store_keys_vii:n #1
2913 {
2914   \bool_if:NF \__enumext_store_columns_vii_bool
2915   {
2916     \regex_match:nnT { \b columns\b } {#1}
2917     {
2918       \int_set_eq:NN
2919       \__enumext_store_columns_vii_int
2920       \__enumext_columns_vii_int
2921       \tl_put_right:Ne \__enumext_store_opt_vii_tl
2922       {
2923         columns = \exp_not:V \__enumext_store_columns_vii_int ,
2924       }
2925     }
2926   }
2927   \bool_if:NF \__enumext_store_columns_sep_vii_bool
2928   {
2929     \regex_match:nnT { \b columns-sep\b } {#1}
2930     {
2931       \dim_set_eq:NN
2932       \__enumext_store_columns_sep_vii_dim
2933       \__enumext_columns_sep_vii_dim
2934       \tl_put_right:Ne \__enumext_store_opt_vii_tl
2935       {
2936         columns-sep = \exp_not:V \__enumext_store_columns_sep_vii_dim,
2937       }
2938     }
2939   }
2940 }
```

(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`.

```
2941 \cs_new_protected:Nn \__enumext_before_list_vii:
2942 {
2943   \__enumext_vspace_above_vii:
2944   \__enumext_before_args_exec_vii:
2945   \__enumext_start_mini_vii:
2946 }
```

(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.

```

2947 \cs_new_protected:Nn \__enumext_after_list_vii:
2948 {
2949   \__enumext_stop_mini_vii:
2950   \__enumext_after_stop_list_vii:
2951   \__enumext_vspace_below_vii:
2952   \int_gset_eq:NN \g__enumext_resume_vii_int \value{enumXvii}
2953   \int_if_exist:cT { g__enumext_resume_ \l__enumext_store_name_tl _int }
2954   {
2955     \int_gset_eq:cN
2956       { g__enumext_resume_ \l__enumext_store_name_tl _int }
2957       { \value{enumXvii} }
2958   }
2959   \bool_lazy_and:nnT { \g__enumext_starred_bool } { \l__enumext_check_ans_bool }
2960   {
2961     \bool_gset_true:N \g__enumext_check_ans_show_h_bool
2962     \tl_gset:NV \g__enumext_store_name_tl \l__enumext_store_name_tl
2963   }
2964   \bool_gset_false:N \g__enumext_starred_bool
2965   \bool_set_false:N \l__enumext_starred_bool
2966 }

```

(End of definition for `__enumext_after_list_vii:`.)

`__enumext_start_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`.

`__enumext_stop_store_level_vii:`

```

2967 \cs_new_protected:Nn \__enumext_start_store_level_vii:
2968 {
2969   \bool_if:NT \l__enumext_store_active_bool
2970   {
2971     \int_compare:nNnT { \l__enumext_level_int } > { \c_zero_int }
2972     {
2973       \__enumext_store_level_open_vii:
2974     }
2975   }
2976 }
2977 \cs_new_protected:Nn \__enumext_stop_store_level_vii:
2978 {
2979   \bool_if:NT \l__enumext_store_active_bool
2980   {
2981     \int_compare:nNnT { \l__enumext_level_int } > { \c_zero_int }
2982     {
2983       \__enumext_store_level_close_vii:
2984     }
2985   }
2986 }

```

(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`.

```

2987 \cs_new_protected_nopar:Nn \__enumext_start_item_tmp_vii:
2988 {
2989   \__enumext_stop_item_tmp_vii:
2990   \int_incr:N \l__enumext_item_column_pos_vii_int
2991   \int_gincr:N \g__enumext_item_count_all_vii_int
2992   \__enumext_item_peek_args_vii:
2993 }

```

(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).

```

2994 \cs_new_protected:Nn \__enumext_item_peek_args_vii:
2995 {
2996   \peek_meaning:NTF (
2997     { \__enumext_joined_item_vii:w }
2998     { \__enumext_joined_item_vii:w (1) }
2999   }

```

(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`.

```

3000 \cs_new_protected:Npn \__enumext_joined_item_vii:w (#1)
3001 {
3002   \__enumext_starred_joined_item_vii:n {#1}
3003   \peek_meaning_remove:NTF *
3004     { \__enumext_starred_item_vii:w }
3005     { \__enumext_standard_item_vii:w }
3006 }

```

(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]`.

```

3007 \cs_new_protected:Npn \__enumext_standard_item_vii:w
3008 {
3009   \bool_set_false:N \l__enumext_item_starred_vii_bool
3010   \peek_meaning:NTF [
3011     {
3012       \bool_set_eq:NN
3013         \l__enumext_wrap_label_vii_bool
3014         \l__enumext_wrap_label_opt_vii_bool
3015       \__enumext_start_item_vii:w
3016     }
3017     {
3018       \bool_set_true:N \l__enumext_wrap_label_vii_bool
3019       \legacy_if_set_true:n { @noitemarg }
3020       \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
3021     }
3022   }

```

(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>]`.

```

3023 \cs_new_protected:Npn \__enumext_starred_item_vii:w
3024 {
3025   \bool_set_true:N \l__enumext_item_starred_vii_bool
3026   \bool_set_true:N \l__enumext_wrap_label_vii_bool
3027   \peek_meaning:NTF [
3028     { \__enumext_starred_item_vii_aux_i:w }
3029     { \__enumext_starred_item_vii_aux_ii:w }
3030   }
3031   \cs_new_protected:Npn \__enumext_starred_item_vii_aux_i:w [#1]
3032   {
3033     \tl_gset:Nn \g__enumext_item_symbol_aux_vii_tl {#1}
3034     \__enumext_starred_item_vii_aux_ii:w
3035   }
3036   \cs_new_protected:Npn \__enumext_starred_item_vii_aux_ii:w

```

```

3037 {
3038   \peek_meaning:NTF [
3039     { \__enumext_starred_item_vii_aux_iii:w }
3040     {
3041       \dim_set_eq:NN
3042       \l__enumext_item_symbol_sep_vii_dim
3043       \l__enumext_labelsep_vii_dim
3044       \legacy_if_set_true:n { @noitemarg }
3045       \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
3046     }
3047   }
3048   \cs_new_protected:Npn \__enumext_starred_item_vii_aux_iii:w [#1]
3049   {
3050     \dim_set:Nn \l__enumext_item_symbol_sep_vii_dim {#1}
3051     \legacy_if_set_true:n { @noitemarg }
3052     \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
3053   }

```

(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.

```

3054 \cs_new_protected_nopar:Npn \__enumext_start_item_vii:w [#1]
3055 {
3056   \cs_set_eq:NN \__enumext_stop_item_tmp_vii: \__enumext_stop_item_vii:
3057   \legacy_if:nT { @noitemarg }
3058   {
3059     \legacy_if_set_false:n { @noitemarg }
3060     \legacy_if:nT { @nmbrrlist }
3061     {
3062       \bool_if:NT \l__enumext_hyperref_bool
3063       {
3064         \legacy_if_set_true:n { @hyper@item }
3065       }
3066       \refstepcounter{enumXvii}
3067       % code for check-ans
3068       \bool_if:NT \l__enumext_check_ans_bool
3069       {
3070         % If true |no-store| key => nested in |enumext|
3071         \bool_if:NTF \l__enumext_store_ans_bool
3072         {
3073           \int_gadd:cn { g__enumext_count_item_ \__enumext_level: _int }
3074           { \int_use:c { g__enumext_count_level_ \__enumext_level: _int } + 1 }
3075         }
3076         {
3077           \int_gincr:N \g__enumext_count_item_all_int
3078           \int_gincr:N \g__enumext_count_level_vii_int
3079         }
3080       }
3081     }
3082   }

```

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.

```

3083 \group_begin:
3084 \lrbox{ \l__enumext_item_text_vii_box }
3085 \bool_if:NF \l__enumext_footnotes_key_bool
3086 {

```

```

3087     \__enumext_renew_footnote:
3088   }
3089   \bool_if:NT \l__enumext_item_starred_vii_bool
3090   {
3091     \tl_if_blank:VT \g__enumext_item_symbol_aux_vii_tl
3092     {
3093       \tl_gset_eq:NN
3094         \g__enumext_item_symbol_aux_vii_tl \l__enumext_item_symbol_vii_tl
3095     }
3096     \mode_leave_vertical:
3097     \skip_horizontal:n { -\l__enumext_item_symbol_sep_vii_dim }
3098     \makebox[ \opt ][ r ]{ \g__enumext_item_symbol_aux_vii_tl }
3099     \skip_horizontal:N \l__enumext_item_symbol_sep_vii_dim
3100     \tl_gclear:N \g__enumext_item_symbol_aux_vii_tl
3101   }
3102   \group_begin:
3103     \tl_use:N \l__enumext_label_font_style_vii_tl
3104     \bool_if:NTF \l__enumext_wrap_label_vii_bool
3105     {
3106       \makebox[ \l__enumext_labelwidth_vii_dim ][ \l__enumext_align_label_vii_str ]
3107         { \__enumext_wrapper_label_vii:n {#1} }
3108     }
3109     {
3110       \makebox[ \l__enumext_labelwidth_vii_dim ][ \l__enumext_align_label_vii_str ]{ #1 }
3111     }
3112   \group_end:
3113   \skip_horizontal:N \l__enumext_labelsep_vii_dim
3114   \tl_use:N \l__enumext_after_list_args_vii_tl
3115   \__enumext_minipage:w [ t ]{ \l__enumext_joined_width_vii_dim }
3116     \skip_set_eq:NN \parindent \l__enumext_listparindent_vii_dim
3117     \skip_set_eq:NN \parskip \l__enumext_parsep_vii_skip
3118     \tl_use:N \l__enumext_fake_item_indent_vii_tl
3119 }

```

(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.

```

3120 \cs_new_protected_nopar:Nn \__enumext_stop_item_vii:
3121 {
3122   \__enumext_endminipage:
3123   \endlrbox
3124   \group_end:
3125   \box_set_wd:Nn \l__enumext_item_text_vii_box
3126   {
3127     \l__enumext_joined_width_vii_dim
3128     + \l__enumext_labelwidth_vii_dim
3129     + \l__enumext_labelsep_vii_dim
3130   }
3131   \int_set:Nn \hbadness { 10000 }
3132   \box_use:N \l__enumext_item_text_vii_box
3133   \bool_if:NF \l__enumext_footnotes_key_bool
3134   {
3135     \__enumext_print_footnote:
3136   }
3137   \int_compare:nNnTF { \l__enumext_item_column_pos_vii_int } = { \l__enumext_columns_vii_int }
3138   {
3139     \par\noindent
3140     \int_zero:N \l__enumext_item_column_pos_vii_int
3141   }
3142   { \hspace{ \l__enumext_columns_sep_vii_dim } }
3143 }

```

(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.

```

3144 \cs_new_protected:Nn \__enumext_remove_extra_parsep_vii:
3145 {
3146   \int_compare:nNnT

```



```

3147     {
3148         \int_mod:nn { \g__enumext_item_count_all_vii_int } { \l__enumext_columns_vii_int }
3149     }
3150     =
3151     { \c_zero_int }
3152     {
3153         \par
3154         \vspace{ -\l__enumext_itemsep_vii_skip }
3155         \int_gzero:N \g__enumext_item_count_all_vii_int
3156     }
3157 }

```

(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`.

```

3158 \__enumext_after_env:nn {enumext*}
3159 {
3160     \bool_if:NT \g__enumext_check_ans_show_h_bool
3161     {
3162         \int_compare:nNnT { \l__enumext_level_int } = { 0 }
3163         {
3164             \__enumext_check_ans_active_vii:
3165         }
3166     }
3167     \bool_gset_false:N \g__enumext_check_ans_show_h_bool
3168     \tl_gclear:N \g__enumext_store_name_tl
3169 }

```

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.

```

3170 \cs_new_protected:Nn \__enumext_starred_columns_set_viii:
3171 {
3172     \dim_compare:nNnT { \l__enumext_columns_sep_viii_dim } = { \c_zero_dim }
3173     {
3174         \dim_set:Nn \l__enumext_columns_sep_viii_dim
3175         {
3176             ( \l__enumext_labelwidth_viii_dim + \l__enumext_labelsep_viii_dim )
3177             / \l__enumext_columns_viii_int
3178         }
3179     }
3180     \int_set:Nn \l__enumext_tmpa_viii_int { \l__enumext_columns_viii_int - \c_one_int }
3181     \dim_set:Nn \l__enumext_item_width_viii_dim
3182     {
3183         ( \linewidth - \l__enumext_columns_sep_viii_dim * \l__enumext_tmpa_viii_int )
3184         / \l__enumext_columns_viii_int - \l__enumext_labelwidth_viii_dim
3185         - \l__enumext_labelsep_viii_dim
3186     }
3187     \dim_zero_new:N \itemwidth
3188 }

```

(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`.

```

3189 \cs_new_protected:Npn \__enumext_starred_joined_item_viii:n #1
3190 {
3191     \int_set:Nn \l__enumext_joined_item_viii_int {#1}
3192     \int_compare:nNnT { \l__enumext_joined_item_viii_int } > { \l__enumext_columns_viii_int }
3193     {
3194         \msg_warning:nnee { enumext } { item-joined }
3195         { \int_use:N \l__enumext_joined_item_viii_int }
3196         { \int_use:N \l__enumext_columns_viii_int }
3197         \int_set:Nn \l__enumext_joined_item_viii_int
3198         {
3199             \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + \c_one_int
3200         }
3201     }

```

```

3202 \int_compare:nNtT
3203 { \l__enumext_joined_item_viii_int }
3204 >
3205 { \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + \c_one_int }
3206 {
3207   \msg_warning:nnee { enumext } { item-joined-columns }
3208   { \int_use:N \l__enumext_joined_item_viii_int }
3209   {
3210     \int_eval:n
3211     { \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + \c_one_int }
3212   }
3213   \int_set:Nn \l__enumext_joined_item_viii_int
3214   {
3215     \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + \c_one_int
3216   }
3217 }
3218 \int_compare:nNtF { \l__enumext_joined_item_viii_int } > { \c_one_int }
3219 {
3220   \int_set_eq:NN \l__enumext_joined_item_aux_viii_int \l__enumext_joined_item_viii_int
3221   \int_decr:N \l__enumext_joined_item_aux_viii_int
3222   \int_add:Nn \l__enumext_item_column_pos_viii_int { \l__enumext_joined_item_aux_viii_int }
3223   \int_gadd:Nn \g__enumext_item_count_all_viii_int { \l__enumext_joined_item_aux_viii_int }
3224   \dim_set:Nn \l__enumext_joined_width_viii_dim
3225   {
3226     \l__enumext_item_width_viii_dim * \l__enumext_joined_item_viii_int
3227     + ( \l__enumext_labelwidth_viii_dim + \l__enumext_labelsep_viii_dim
3228         + \l__enumext_columns_sep_viii_dim
3229         ) * \l__enumext_joined_item_aux_viii_int
3230   }
3231   \dim_set_eq:NN \itemwidth \l__enumext_joined_width_viii_dim
3232 }
3233 {
3234   \dim_set_eq:NN \l__enumext_joined_width_viii_dim \l__enumext_item_width_viii_dim
3235   \dim_set_eq:NN \itemwidth \l__enumext_item_width_viii_dim
3236 }
3237 }

```

(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.

```

3238 \cs_new_protected:Nn \__enumext_start_mini_viii:
3239 {
3240   \dim_compare:nNtT { \l__enumext_minipage_right_viii_dim } > { \c_zero_dim }
3241   {
3242     \dim_set:Nn \l__enumext_minipage_left_viii_dim
3243     {
3244       \linewidth
3245       - \l__enumext_minipage_right_viii_dim
3246       - \l__enumext_minipage_hsep_viii_dim
3247     }
3248     \bool_set_true:N \l__enumext_minipage_active_viii_bool
3249     \dim_gset_eq:NN
3250     \g__enumext_minipage_right_viii_dim
3251     \l__enumext_minipage_right_viii_dim
3252     \__enumext_mini_addvspace_viii:
3253     \nointerlineskip\noindent
3254     \begin{__enumext_mini_env*}{ \l__enumext_minipage_left_viii_dim }
3255   }
3256 }
3257 \cs_new_protected:Nn \__enumext_stop_mini_viii:
3258 {
3259   \bool_if:NT \l__enumext_minipage_active_viii_bool
3260   {
3261     \end{__enumext_mini_env*}
3262     \hfill
3263     \bool_gset_true:N \g__enumext_minipage_active_viii_bool
3264   }
3265 }
3266 \__enumext_after_env:nn {keyans*}
3267 {
3268   \bool_if:NT \g__enumext_minipage_active_viii_bool

```

```

3269     {
3270         \begin{__enumext_mini_env*}{ \g__enumext_minipage_right_viii_dim }
3271         \par\addvspace { \g__enumext_minipage_right_skip }
3272         \bool_if:NF \g__enumext_minipage_center_viii_bool
3273         {
3274             \centering
3275         }
3276         \tl_use:N \g__enumext_miniright_code_viii_tl % the code
3277         \end{__enumext_mini_env*}
3278         \par\addvspace{ \g__enumext_minipage_after_skip }
3279     }
3280     \bool_gset_false:N \g__enumext_minipage_active_viii_bool
3281     \bool_gset_true:N \g__enumext_minipage_center_viii_bool
3282     \tl_gclear:N \g__enumext_miniright_code_viii_tl
3283     \dim_gzero:N \g__enumext_minipage_right_viii_dim
3284 }

```

(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.

```

3285 \NewDocumentEnvironment{keyans*}{ o }
3286 {
3287     \__enumext_safe_exec_viii:
3288     \__enumext_parse_keys_viii:n {#1}
3289     \__enumext_before_list_viii:
3290     \__enumext_start_list:nn { }
3291     {
3292         \__enumext_list_arg_two_viii:
3293         \__enumext_before_keys_exec_viii:
3294     }
3295     \__enumext_starred_columns_set_viii:
3296     \item[] \scan_stop:
3297     \cs_set_eq:NN \__enumext_stop_item_tmp_viii: \noindent
3298     \cs_set_eq:NN \item \__enumext_start_item_tmp_viii:
3299 }
3300 {
3301     \__enumext_stop_item_tmp_viii:
3302     \__enumext_remove_extra_parsep_viii:
3303     \__enumext_stop_list:
3304     \__enumext_after_list_viii:
3305 }

```

(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.

```

3306 \cs_new_protected:Nn \__enumext_safe_exec_viii:
3307 {
3308     \int_incr:N \l__enumext_keyans_level_h_int
3309     \int_compare:nNnT { \l__enumext_keyans_level_h_int } > { 1 }
3310     {
3311         \msg_error:nn { enumext } { nested }
3312     }
3313     % Set false for interfering with enumext nested in keyans* (yes, its possible and crayze)
3314     \bool_set_false:N \l__enumext_store_active_bool
3315     \int_compare:nNnT { \l__enumext_level_int } > { 1 }
3316     {
3317         \msg_error:nn { enumext } { keyans-wrong-level }
3318     }
3319 }

```

(End of definition for __enumext_safe_exec_viii:.)

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

```

3320 \cs_new_protected:Npn \__enumext_parse_keys_viii:n #1
3321 {
3322     \tl_if_novalue:nF {#1}
3323     {
3324         \keys_set:nn { enumext / keyans* } {#1}
3325     }
3326 }

```

(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`.

```
3327 \cs_new_protected:Nn \__enumext_before_list_viii:
3328 {
3329     \__enumext_vspace_above_viii:
3330     \__enumext_before_args_exec_viii:
3331     \__enumext_start_mini_viii:
3332 }
```

(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.

```
3333 \cs_new_protected:Nn \__enumext_after_list_viii:
3334 {
3335     \__enumext_stop_mini_viii:
3336     \__enumext_after_stop_list_viii:
3337     \__enumext_vspace_below_viii:
3338 }
```

(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`.

```
3339 \cs_new_protected_nopar:Nn \__enumext_start_item_tmp_viii:
3340 {
3341     \__enumext_stop_item_tmp_viii:
3342     \int_incr:N \__enumext_item_column_pos_viii_int
3343     \int_gincr:N \g__enumext_item_count_all_viii_int
3344     \__enumext_item_peek_args_viii:
3345 }
```

(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).

```
3346 \cs_new_protected:Nn \__enumext_item_peek_args_viii:
3347 {
3348     \peek_meaning:NTF (
3349         { \__enumext_joined_item_viii:w }
3350         { \__enumext_joined_item_viii:w (1) }
3351     }
```

(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`.

```
3352 \cs_new_protected:Npn \__enumext_joined_item_viii:w (#1)
3353 {
3354     \__enumext_starred_joined_item_viii:n {#1}
3355     \peek_meaning_remove:NTF *
3356         { \__enumext_starred_item_viii:w }
3357         { \__enumext_standard_item_viii:w }
3358 }
```

(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]`.

```

3359 \cs_new_protected:Npn \__enumext_standard_item_viii:w
3360 {
3361   \bool_set_false:N \l__enumext_item_starred_viii_bool
3362   \peek_meaning:NTF [
3363     {
3364       \bool_set_eq:NN
3365         \l__enumext_wrap_label_viii_bool
3366         \l__enumext_wrap_label_opt_viii_bool
3367       \__enumext_start_item_viii:w
3368     }
3369     {
3370       \bool_set_true:N \l__enumext_wrap_label_viii_bool
3371       \legacy_if_set_true:n { @noitemarg }
3372       \__enumext_start_item_viii:w [ \l__enumext_label_viii_tl ]
3373     }
3374   }

```

(End of definition for `__enumext_standard_item_viii:w`)

`__enumext_starred_item_viii: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⟩]`.

`__enumext_starred_item_viii_aux_i:w`

`__enumext_starred_item_viii_aux_ii:w`

```

3375 \cs_new_protected:Npn \__enumext_starred_item_viii:w
3376 {
3377   \bool_set_true:N \l__enumext_item_starred_viii_bool
3378   \bool_set_true:N \l__enumext_wrap_label_viii_bool
3379   \peek_meaning:NTF [
3380     { \__enumext_starred_item_viii_aux_i:w }
3381     { \__enumext_starred_item_viii_aux_ii:w }
3382   }
3383   \cs_new_protected:Npn \__enumext_starred_item_viii_aux_i:w [#1]
3384   {
3385     \tl_clear:N \l__enumext_store_keyans_label_tl
3386     \tl_if_novalue:nF { #1 }
3387     {
3388       \tl_set:Nx \l__enumext_keyans_tmpa_tl { \c_space_tl [#1] }
3389       \tl_set:Nx \l__enumext_keyans_tmpb_tl { \c_space_tl #1 }
3390     }
3391     \__enumext_starred_item_viii_aux_ii:w
3392   }
3393   \cs_new_protected:Npn \__enumext_starred_item_viii_aux_ii:w
3394   {
3395     \legacy_if_set_true:n { @noitemarg }
3396     \__enumext_start_item_viii:w [ \l__enumext_label_viii_tl ]
3397   }

```

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

Pass content to prop list and more

`__enumext_starred_item_exec_viii:`

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

```

3398 \cs_new_protected:Nn \__enumext_starred_item_exec:
3399 {
3400   \tl_put_left:Nx \l__enumext_store_keyans_label_tl { \l__enumext_label_viii_tl }
3401   \tl_if_blank:VF \l__enumext_keyans_tmpb_tl
3402   {
3403     \tl_put_right:Nx \l__enumext_store_keyans_label_tl { \l__enumext_keyans_tmpb_tl }
3404   }
3405   \__enumext_store_addto_prop:V \l__enumext_store_keyans_label_tl
3406   \__enumext_keyans_store_ref:

```

```

3407 \tl_put_left:Ne \l__enumext_store_keyans_label_tl { \item }
3408 \__enumext_keyans_addto_seq_link:
3409 \bool_if:NT \l__enumext_show_answer_bool
3410 {
3411   \tl_if_blank:VF \l__enumext_keyans_tmpa_tl
3412   {
3413     \tl_put_right:Ne \l__enumext_label_viii_tl { \l__enumext_keyans_tmpa_tl }
3414     \__enumext_label_width_by_box:Nn \l__enumext_keyans_tmpa_dim { \tl_use:N \l__enumext_
3415     \dim_add:Nn \l__enumext_fake_item_indent_viii_dim { \l__enumext_keyans_tmpa_dim }
3416   }
3417   \__enumext_print_keyans_box:NN \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
3418 }
3419 \bool_if:NT \l__enumext_show_position_bool
3420 {
3421   \tl_set:Ne \l__enumext_mark_answer_sym_tl
3422   {
3423     \group_begin:
3424     \exp_not:N \normalfont
3425     \exp_not:N \footnotesize [ \int_eval:n
3426       {
3427         \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
3428       }
3429     ]
3430     \group_end:
3431   }
3432   \tl_if_blank:VF \l__enumext_keyans_tmpa_tl
3433   {
3434     \tl_put_right:Ne \l__enumext_label_viii_tl { \l__enumext_keyans_tmpa_tl }
3435     \__enumext_label_width_by_box:Nn \l__enumext_keyans_tmpa_dim { \tl_use:N \l__enumext_
3436     \dim_add:Nn \l__enumext_fake_item_indent_viii_dim { \l__enumext_keyans_tmpa_dim }
3437   }
3438   \__enumext_print_keyans_box:NN \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
3439 }
3440 }

```

(End of definition for `__enumext_starred_item_exec_viii:`.)

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.

```

3441 \cs_new_protected_nopar:Npn \__enumext_start_item_viii:w [#1]
3442 {
3443   \cs_set_eq:NN \__enumext_stop_item_tmp_viii: \__enumext_stop_item_viii:
3444   \legacy_if:nT { @noitemarg }
3445   {
3446     \legacy_if_set_false:n { @noitemarg }
3447     \legacy_if:nT { @nmbrrlist }
3448     {
3449       \bool_if:NT \l__enumext_hyperref_bool
3450       {
3451         \legacy_if_set_true:n { @hyper@item }
3452       }
3453       \refstepcounter{enumXviii}
3454     }
3455   }

```

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

```

3456 \group_begin:
3457 \lrbox{ \l__enumext_item_text_viii_box }
3458 \bool_if:NF \l__enumext_footnotes_key_bool
3459 {
3460   \__enumext_renew_footnote:
3461 }
3462 \bool_if:NT \l__enumext_item_starred_viii_bool
3463 {

```

```

3464         \__enumext_starred_item_exec:
3465     }
3466     \group_begin:
3467         \tl_use:N \l__enumext_label_font_style_viii_tl
3468         \bool_if:NTF \l__enumext_wrap_label_viii_bool
3469         {
3470             \makebox[ \l__enumext_labelwidth_viii_dim ][ \l__enumext_align_label_viii_str ]
3471                 { \__enumext_wrapper_label_viii:n {#1} }
3472         }
3473         {
3474             \makebox[ \l__enumext_labelwidth_viii_dim ][ \l__enumext_align_label_viii_str ]{ #1
3475         }
3476     \group_end:
3477     \skip_horizontal:N \l__enumext_labelsep_viii_dim
3478     \tl_use:N \l__enumext_after_list_args_viii_tl
3479     \__enumext_minipage:w [ t ]{ \l__enumext_joined_width_viii_dim }
3480     \skip_set_eq:NN \parindent \l__enumext_listparindent_viii_dim
3481     \skip_set_eq:NN \parskip \l__enumext_parsep_viii_skip
3482     \tl_use:N \l__enumext_fake_item_indent_viii_tl
3483 }

```

(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.

```

3484 \cs_new_protected_nopar:Nn \__enumext_stop_item_viii:
3485 {
3486     \__enumext_endminipage:
3487     \endlrbox
3488     \group_end:
3489     \box_set_wd:Nn \l__enumext_item_text_viii_box
3490     {
3491         \l__enumext_joined_width_viii_dim
3492         + \l__enumext_labelwidth_viii_dim
3493         + \l__enumext_labelsep_viii_dim
3494     }
3495     \int_set:Nn \hbadness { 10000 }
3496     \box_use:N \l__enumext_item_text_viii_box
3497     \bool_if:NF \l__enumext_footnotes_key_bool
3498     {
3499         \__enumext_print_footnote:
3500     }
3501     \int_compare:nNnTF { \l__enumext_item_column_pos_viii_int } = { \l__enumext_columns_viii_int }
3502     {
3503         \par\noindent
3504         \int_zero:N \l__enumext_item_column_pos_viii_int
3505     }
3506     { \hspace{ \l__enumext_columns_sep_viii_dim } }
3507 }

```

(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.

```

3508 \cs_new_protected:Nn \__enumext_remove_extra_parsep_viii:
3509 {
3510     \int_compare:nNnT
3511     {
3512         \int_mod:nn { \g__enumext_item_count_all_viii_int } { \l__enumext_columns_viii_int }
3513     }
3514     =
3515     { \c_zero_int }
3516     {
3517         \par
3518         \vspace{ -\l__enumext_itemsep_viii_skip }
3519         \int_gzero:N \g__enumext_item_count_all_viii_int
3520     }
3521 }

```

(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.

```
3522 \NewDocumentCommand \getkeyans { m }
3523 {
3524   \exp_args:Ne \__enumext_getkeyans_aux:n
3525   { \tl_to_str:e { \text_expand:n {#1} } }
3526 }
```

(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.

```
3527 \cs_new_protected:Npn \__enumext_getkeyans_aux:n #1
3528 {
3529   \str_if_in:nnTF {#1} { : }
3530   {
3531     \use:e
3532     {
3533       \cs_set:Npn \exp_not:N \__enumext_tmp:w ##1 \c_colon_str ##2 \scan_stop:
3534       { {##1} {##2} }
3535     }
3536     \exp_after:wN \__enumext_getkeyans:nn \__enumext_tmp:w #1 \scan_stop:
3537   }
3538   { \msg_error:nnn { enumext } { missing-colon } {#1} }
3539 }
```

(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 $\langle prop\ list \rangle$.

```
3540 \cs_new_protected:Npn \__enumext_getkeyans:nn #1 #2
3541 {
3542   \prop_if_exist:cF { g__enumext_#1_prop }
3543   { \msg_error:nnn { enumext } { undefined-storage-anskey } {#1} }
3544   \group_begin:
3545   \prop_item:cn { g__enumext_#1_prop }{#2}
3546   \group_end:
3547 }
```

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

10.38 The command \printkeyans

The `\printkeyans` command prints “all stored content” in the $\langle sequence \rangle$ defined by the `save-ans` key. The first thing we will do is to define a set of $\langle keys \rangle$ with which we will control the options of the different nesting levels for the `enumext` and `enumext*` environment by storing the values of these in the token list variables `\l__enumext_print_keyans_X_tl`.

```
3548 \keys_define:nn { keyanskey / print }
3549 {
3550   level-1 .code:n = \tl_put_right:Nn \l__enumext_print_keyans_i_tl
3551   {
3552     \setenumext[level,1] {#1} \setenumext[print,1] {#1}
3553   },
3554   level-1 .initial:n = { label=\arabic*. , nosep, columns=2, first=\small, font=\small },
3555   level-2 .code:n = \tl_put_right:Nn \l__enumext_print_keyans_ii_tl
3556   {
3557     \setenumext[level,2] {#1} \setenumext[print,2] {#1}
3558   },
3559   level-2 .initial:n = { nosep, label=(\alph*), first=\small, font=\small },
3560   level-3 .code:n = \tl_put_right:Nn \l__enumext_print_keyans_iii_tl
3561   {
3562     \setenumext[level,3] {#1} \setenumext[print,3] {#1}
3563   },
3564   level-3 .initial:n = { nosep, label=\roman*. , first=\small, font=\small },
3565   level-4 .code:n = \tl_put_right:Nn \l__enumext_print_keyans_iv_tl
3566   {
3567     \setenumext[level,4] {#1} \setenumext[print,4] {#1}
3568   },
3569 }
```



```

3569     level-4 .initial:n = { nosep, label=\Alph*., first=\small, font=\small },
3570     level-* .code:n = \tl_put_right:Nn \l__enumext_print_keyans_vii_tl % starred
3571                     {
3572                         \setenumext[enumext*] {#1} %%\setenumext[print,*] {#1}
3573                     },
3574     level-* .initial:n = { label=\arabic*., nosep, columns=2, first=\small, font=\small },
3575 }

```

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

```

3576 \NewDocumentCommand \printkeyans { s O{} m }
3577 {
3578     \group_begin:
3579     \tl_use:N \l__enumext_print_keyans_i_tl
3580     \tl_use:N \l__enumext_print_keyans_ii_tl
3581     \tl_use:N \l__enumext_print_keyans_iii_tl
3582     \tl_use:N \l__enumext_print_keyans_iv_tl
3583     \tl_use:N \l__enumext_print_keyans_vii_tl
3584     \__enumext_printkeyans:nnn { #1 } { #2 } { #3 }
3585     \group_end:
3586 }

```

(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

```

3587 \cs_new_protected:Npn \__enumext_printkeyans:nnn #1 #2 #3
3588 {
3589     \seq_if_exist:cTF { g__enumext_#3_seq }
3590     {
3591         \seq_if_empty:cF { g__enumext_#3_seq }
3592         {
3593             %%\seq_show:c { g__enumext_#3_seq }
3594             \bool_if:nTF {#1}
3595             {
3596                 \begin{enumext*}[#2]
3597                 \seq_map_inline:cn { g__enumext_#3_seq } { ##1 }
3598                 \end{enumext*}
3599             }
3600             {
3601                 \begin{enumext}[#2]
3602                 \seq_map_inline:cn { g__enumext_#3_seq } { ##1 }
3603                 \end{enumext}
3604             }
3605         }
3606     }
3607     {
3608         \msg_error:nnn { enumext } { undefined-storage-anskey } {#3}
3609     }
3610 }

```

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

10.39 The command `\setenumext`

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

```

3611 \keys_define:nn { enumext / meta-families }
3612 {
3613     level-1 .code:n = { \keys_set:nn { enumext / level-1 } {#1} },
3614     level-2 .code:n = { \keys_set:nn { enumext / level-2 } {#1} },
3615     level-3 .code:n = { \keys_set:nn { enumext / level-3 } {#1} },
3616     level-4 .code:n = { \keys_set:nn { enumext / level-4 } {#1} },
3617     keyans .code:n = { \keys_set:nn { enumext / keyans } {#1} },
3618     enumext* .code:n = { \keys_set:nn { enumext / enumext* } {#1} },
3619     keyans* .code:n = { \keys_set:nn { enumext / keyans* } {#1} },
3620     print-1 .code:n = { \keys_set:nn { keyanskey / print } { level-1 = {#1} } },
3621     print-2 .code:n = { \keys_set:nn { keyanskey / print } { level-2 = {#1} } },
3622     print-3 .code:n = { \keys_set:nn { keyanskey / print } { level-3 = {#1} } },

```

```

3623   print-4 .code:n = { \keys_set:nn { keyanskey / print } { level-4 = {#1} } } ,
3624   print-* .code:n = { \keys_set:nn { keyanskey / print } { level-* = {#1} } } ,
3625   unknown .code:n = { \msg_error:nn { enumext } { unknown-key-family } } ,
3626 }

```

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

```

3627 \seq_const_from_clist:Nn \c__enumext_all_families_seq
3628 {
3629   level-1 , level-2 , level-3 , level-4 , keyans, enumext*,
3630   keyans* , print-1 , print-2 , print-3 , print-4 , print-*,
3631 }

```

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

```

3632 \NewDocumentCommand \setenumext { o +m }
3633 {
3634   \tl_if_novalue:nTF {#1}
3635   {
3636     \seq_map_inline:Nn \c__enumext_all_families_seq
3637   }
3638   {
3639     \seq_clear:N \l__enumext_setkey_tmpa_seq
3640     \seq_set_from_clist:Nn \l__enumext_setkey_tmpb_seq {#1}
3641     \int_set:Nn \l__enumext_setkey_tmpa_int
3642     {
3643       \seq_count:N \l__enumext_setkey_tmpb_seq
3644     }
3645     \int_compare:nNnTF { \l__enumext_setkey_tmpa_int } > { 1 }
3646     {
3647       \seq_pop_left:NN \l__enumext_setkey_tmpb_seq \l__enumext_setkey_tmpa_tl
3648       \seq_map_function:NN \l__enumext_setkey_tmpb_seq \l__enumext_set_parse:n
3649       \seq_set_map_e:NNn \l__enumext_setkey_tmpa_seq \l__enumext_setkey_tmpa_seq
3650       {
3651         \tl_use:N \l__enumext_setkey_tmpa_tl - ##1
3652       }
3653     }
3654     {
3655       \seq_put_right:Ne \l__enumext_setkey_tmpa_seq { \tl_trim_spaces:n {#1} }
3656     }
3657     \seq_if_empty:NTF \l__enumext_setkey_tmpa_seq
3658     { \seq_map_inline:Nn \c__enumext_all_families_seq }
3659     { \seq_map_inline:Nn \l__enumext_setkey_tmpa_seq }
3660   }
3661   {
3662     \keys_set:nn { enumext / meta-families } { ##1 = {#2} }
3663   }
3664 }

```

(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.

```

3665 \cs_new_protected:Npn \__enumext_set_parse:n #1
3666 {
3667   \tl_set:Ne \l__enumext_setkey_tmpb_tl { \tl_trim_spaces:n {#1} }
3668   \int_step_inline:nnn { 0 } { 4 } % <- max level
3669   { \tl_remove_all:Nn \l__enumext_setkey_tmpb_tl {##1} }
3670   \tl_if_empty:NTF \l__enumext_setkey_tmpb_tl
3671   {
3672     \seq_put_right:Ne \l__enumext_setkey_tmpa_seq
3673     { \tl_trim_spaces:n {#1} }
3674   }
3675   { \__enumext_set_error:nn {#1} { } }
3676 }
3677 \cs_new_protected:Npn \__enumext_set_error:nn #1 #2
3678 { \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.

```

3679 \msg_new:nnn { enumext } { package-load }
3680 {
3681   The ~ '#1' ~ package ~ is ~ already ~ loaded.
3682 }

3683 \msg_new:nnn { enumext } { package-not-load }
3684 {
3685   The ~ '#1' ~ package ~ will ~ be ~ loaded ~ as ~ a ~ dependency.
3686 }

3687 \msg_new:nnn { enumext } { package-load-foot }
3688 {
3689   The ~ '#1' ~ package ~ is ~ loaded ~ with ~ the ~ option ~ '#2'.
3690 }
```

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

```

3691 \msg_new:nnn { enumext } { counters }
3692 {
3693   The ~ counter ~ '#1' ~ is ~ already ~ defined ~ by ~ some ~ \\
3694   package ~ or ~ macro, ~ it ~ cannot ~ be ~ continued.
3695 }
```

Message used by [*(key = val)*] system and **\setenumext** command.

```

3696 \msg_new:nnn { enumext } { invalid-key }
3697 {
3698   The ~ key ~ '#1' ~ is ~ not ~ know ~ the ~ level ~ #2.
3699 }
3700 \msg_new:nnn { enumext } { unknown-key-family }
3701 {
3702   Unknown~key~family~`\l_keys_key_str'~for~enumext.
3703 }
```

Messages used in length calculation.

```

3704 \msg_new:nnn { enumext } { width-negative }
3705 {
3706   Ignoring ~ negative ~ value ~ '#1=#2' ~ \msg_line_context:.\
3707   The ~ key ~ '#1'~ accepts ~ values ~ >= ~ 0pt.
3708 }
3709 \msg_new:nnn { enumext } { width-zero }
3710 {
3711   Invalid ~ '#1=#2' ~ \msg_line_context:.\
3712   The ~ key ~ '#1'~ accepts ~ values ~ > ~ 0pt.
3713 }
```

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

```

3714 \msg_new:nnn { enumext } { list-lengths }
3715 {
3716   **** ~ Lengths ~ used ~ by ~ 'enumext' ~ level ~ '#2' ~ \msg_line_context:~\c_space_tl ****\
3717   \__enumext_show_length:nnn { dim } { labelsep } { #1}
3718   \__enumext_show_length:nnn { dim } { labelwidth } { #1}
3719   \__enumext_show_length:nnn { dim } { itemindent } { #1}
3720   \__enumext_show_length:nnn { dim } { leftmargin } { #1}
3721   \__enumext_show_length:nnn { dim } { rightmargin } { #1}
3722   \__enumext_show_length:nnn { dim } { listparindent } { #1}
3723   \__enumext_show_length:nnn { skip } { topsep } { #1}
3724   \__enumext_show_length:nnn { skip } { parsep } { #1}
3725   \__enumext_show_length:nnn { skip } { partopsep } { #1}
3726   \__enumext_show_length:nnn { skip } { itemsep } { #1}
3727   ****
3728 }
```

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

```

3729 \msg_new:nnn { enumext } { list-lengths-not-nested }
3730 {
3731   **** ~ Lengths ~ used ~ by ~ '#2' ~ environment ~ \msg_line_context:~\c_space_tl ****\
3732   \__enumext_show_length:nnn { dim } { labelsep } { #1}
3733   \__enumext_show_length:nnn { dim } { labelwidth } { #1}
3734   \__enumext_show_length:nnn { dim } { itemindent } { #1}
3735   \__enumext_show_length:nnn { dim } { leftmargin } { #1}
3736   \__enumext_show_length:nnn { dim } { rightmargin } { #1}
3737   \__enumext_show_length:nnn { dim } { listparindent } { #1}
```

```

3738     \__enumext_show_length:nnn { skip } { topsep } {#1}
3739     \__enumext_show_length:nnn { skip } { parsep } {#1}
3740     \__enumext_show_length:nnn { skip } { partopsep } {#1}
3741     \__enumext_show_length:nnn { skip } { itemsep } {#1}
3742     *****
3743 }

```

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

```

3744 \msg_new:nnn { enumext } { items-same-answer }
3745 {
3746     *****~Checking~answers~on~'#1'~OK~*****\\
3747     **~ All ~ items ~ stored ~ in ~ sequence ~ '#1' ~ have ~ an ~ answer. \\
3748     *****
3749     \prg_replicate:nn { 7 + \str_count:n {#1} } { * }
3750 }
3751 \msg_new:nnn { enumext } { item-different-answer }
3752 {
3753     Number ~ of ~ items ~ different ~ of ~ number ~ of ~
3754     answer ~ in ~ sequence ~ '#1'~ closed ~ \msg_line_context:.
3755 }

```

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

```

3756 \msg_new:nnn { enumext } { missing-starred }
3757 {
3758     Missing ~ '\c_backslash_str #1*' ~ in ~ '#2' ~ \msg_line_context:.
3759 }

```

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

```

3760 \msg_new:nnn { enumext } { list-too-deep }
3761 {
3762     Too ~ deep ~ nesting ~ for ~ 'enumext' ~ \msg_line_context::~ \\
3763     The ~ maximum ~ level ~ of ~ nesting ~ is ~ 4.
3764 }

```

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

```

3765 \msg_new:nnn { enumext } { anskey-wrong-place }
3766 {
3767     Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context::~ \\
3768     '\c_backslash_str #1' ~ works ~ in ~ the ~ environment ~ '#2'.
3769 }
3770 \msg_new:nnn { enumext } { anspic-wrong-place }
3771 {
3772     Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context::~ \\
3773     '\c_backslash_str #1' ~ works ~ in ~ the ~ environment ~ '#2'.
3774 }
3775 \msg_new:nnn { enumext } { command-wrong-place }
3776 {
3777     Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context::~ \\
3778     '\c_backslash_str #1' ~ works ~ outside ~ the ~ environment ~ '#2'.
3779 }

```

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

```

3780 \msg_new:nnn { enumext } { keyans-nested }
3781 {
3782     The ~ environment ~ 'keyans' ~ can't ~ be ~ nested ~ \msg_line_context:.
3783 }
3784 \msg_new:nnn { enumext } { keyans-wrong-level }
3785 {
3786     Wrong ~ level ~ position ~ for ~ 'keyans' ~ \msg_line_context::~ \\
3787     The ~ environment ~ 'keyans' ~ can ~ only ~ be ~ in ~ the ~ first ~ level.
3788 }
3789 \msg_new:nnn { enumext } { wrong-place }
3790 {
3791     Wrong ~ place ~ for ~ '#1' ~ environment ~ \msg_line_context::~ \\
3792     '#1' ~ is ~ only ~ found ~ with ~ '#2' ~ in ~ 'enumext'.
3793 }
3794 \msg_new:nnn { enumext } { keyanspic-nested }
3795 {
3796     The ~ environment ~ 'keyanspic' ~ can't ~ be ~ nested ~ \msg_line_context::~.
3797 }
3798 \msg_new:nnn { enumext } { keyanspic-wrong-level }
3799 {
3800     Wrong ~ level ~ position ~ for ~ 'keyanspic' ~ \msg_line_context::~ \\

```

```

3801     The ~ environment ~ 'keyans' ~ can ~ only ~ be ~ in ~ the ~ first ~ level.
3802 }

```

Messages used by `\getkeyans` command.

```

3803 \msg_new:nnn { enumext } { undefined-storage-anskey }
3804 {
3805     Storage ~ named ~ '#1' ~ is ~ not ~ defined ~ \msg_line_context:.
3806 }

```

Messages used by `\miniright` command.

```

3807 \msg_new:nnn { enumext } { missing-miniright }
3808 {
3809     Missing ~ '\c_backslash_str miniright' ~ in ~ \msg_line_context:.\
3810     The ~ key ~ 'mini-env' ~ need ~ '\c_backslash_str miniright'.
3811 }
3812 \msg_new:nnn { enumext } { wrong-miniright-place }
3813 {
3814     Wrong ~ place ~ for ~ '\c_backslash_str miniright' ~ \msg_line_context:~ \
3815     Works ~ in ~ 'enumext' ~ and ~ 'keyans' ~ with ~ key ~ 'mini-env'.
3816 }
3817 \msg_new:nnn { enumext } { wrong-miniright-use }
3818 {
3819     Wrong ~ use ~ for ~ '\c_backslash_str miniright' ~ \msg_line_context:~ \
3820     '\c_backslash_str miniright' ~ need ~ a ~ key ~ 'mini-env'.
3821 }

```

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

```

3822 \msg_new:nnn { enumext } { nested }
3823 {
3824     The ~ starred ~ environment ~ can't ~ be ~ nested ~ \msg_line_context:.
3825 }
3826 \msg_new:nnn { enumext } { item-joined }
3827 {
3828     Items ~ joined ~ (#1) ~ > ~ #2 ~ columns ~ \msg_line_context:.
3829 }
3830 \msg_new:nnn { enumext } { item-joined-columns }
3831 {
3832     Not ~ space ~ to ~ join ~ items ~ (#1) ~ > ~ #2 ~ \msg_line_context:.
3833 }

```

10.41 Finish package

Finish package implementation.

```

3834 \file_input_stop:
3835 </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>	402
<code>\+</code>	203
<code>\-</code>	203
<code>\\</code>	211, 2652, 3693, 3706, 3711, 3716, 3731, 3746, 3747, 3762, 3767, 3772, 3777, 3786, 3791, 3800, 3809, 3814, 3819
A	
<code>above</code>	<u>1213</u>
<code>above*</code>	<u>1213</u>
<code>\addvspace</code> ..	860, 888, 1011, 1090, 1153, 1159, 1187, 1204, 2469, 2491, 2608, 2623, 2847, 2854, 3271, 3278
<code>after</code>	<u>698</u>
<code>align</code>	<u>356</u>
<code>\Alph</code>	29, 33, 34
<code>\Alph</code>	308, 485, 503, 516, 3569
<code>\alph</code>	29, 33, 34
<code>\alph</code>	309, 483, 3559
<code>\anskey</code>	10, 57, <u>1647</u>
<code>\anspic</code>	12, 78, <u>2630</u>
<code>\arabic</code>	29, 31
<code>\arabic</code>	307, 482, 502, 3554, 3574
B	
<code>\b</code>	2358, 2371, 2916, 2929
<code>\baselineskip</code>	41
<code>\baselineskip</code>	1607, 1615
<code>before</code>	<u>698</u>
<code>before*</code>	<u>698</u>
<code>below</code>	<u>1213</u>
<code>below*</code>	<u>1213</u>
bool commands:	
<code>\bool_gset_false:N</code> ..	2514, 2856, 2964, 3167, 3280
<code>\bool_gset_true:N</code> 802, 2342, 2476, 2839, 2857, 2898, 2961, 3263, 3281	
<code>\bool_if:NTF</code> .	248, 260, 277, 1235, 1249, 1262, 1273, 1284, 1295, 1306, 1317, 1356, 1363, 1374, 1435, 1437, 1569, 1593, 1600, 1628, 1659, 1672, 1674, 1685, 1705, 1830, 1841, 1845, 1879, 1894, 1963, 1978, 1989, 2065, 2096, 2170, 2186, 2254, 2264, 2300, 2305, 2349, 2356, 2369, 2403, 2453, 2467, 2482, 2507, 2537, 2593, 2606, 2614, 2632, 2835, 2844, 2848, 2906, 2914, 2927, 2969, 2979, 3062, 3068, 3071, 3085, 3089, 3104, 3133, 3160, 3259, 3268, 3272, 3409, 3419, 3449, 3458, 3462, 3468, 3497
<code>\bool_if:nTF</code> 1188, 1205, 1713, 2108, 2142, 2206, 2653, 3594	
<code>\bool_if_p:N</code>	2243, 2290
<code>\bool_lazy_all:nTF</code> ..	1415, 1762, 1771, 1784, 1800, 2336, 2892
<code>\bool_lazy_and:nnTF</code> .	1695, 1738, 1948, 2241, 2289, 2385, 2472, 2959
<code>\bool_lazy_or:nnTF</code>	2658
<code>\bool_new:N</code> 25, 26, 27, 28, 29, 35, 37, 46, 67, 72, 73, 78, 79, 82, 100, 102, 104, 107, 108, 117, 118, 119, 120, 131, 132, 157, 168, 170	
<code>\bool_not_p:n</code> 1697, 1789, 1804, 2338, 2387, 2474, 2894	
<code>\bool_set_eq:NN</code>	2074, 2123, 3012, 3364
<code>\bool_set_false:N</code> 257, 1390, 1391, 1496, 1499, 1519, 1522, 2496, 2544, 2626, 2690, 2707, 2965, 3009, 3314, 3361	
<code>\bool_set_true:N</code> 239, 243, 349, 626, 1219, 1224, 1351, 1370, 1381, 1495, 1498, 1518, 1521, 1531, 1538, 2070, 2101, 2119, 2131, 2335, 2391, 2396, 2422, 2542, 2568, 2824, 2891, 3018, 3025, 3026, 3248, 3370, 3377, 3378	
box commands:	
<code>\box_dp:N</code> .	907, 911, 915, 926, 930, 941, 950, 956, 966, 979, 985, 991, 1022, 1023, 1024, 1027, 1037, 1041, 1050, 1057, 1062, 1070, 1099, 1100, 1103, 1110, 1123, 1131, 1137, 1145, 2719
<code>\box_new:N</code>	43, 163
<code>\box_set_wd:Nn</code>	3125, 3489
<code>\box_use:N</code>	3132, 3496
<code>\box_wd:N</code>	315
C	
<code>\c</code>	402, 403, 526, 528, 540, 542
<code>\cB</code>	403
<code>\cE</code>	403
<code>\centering</code>	1190, 1207, 2740, 2850, 3274
<code>check-ans</code>	<u>1383</u>
Document class:	
<code>article</code>	35
clist commands:	
<code>\clist_const:Nn</code>	175
<code>\clist_map_function:nN</code>	2727
<code>\clist_map_inline:Nn</code> .	355, 568, 631, 697, 712, 793, 1229
<code>\clist_map_inline:nn</code> .	34, 51, 57, 69, 81, 106, 130, 140, 154, 174, 219, 380, 397, 636, 808, 1396, 1508, 1526, 1547, 1759, 1888, 2023, 2235, 2238, 2271, 2281, 2284, 2310
<code>\columnbreak</code>	58
<code>\columnbreak</code>	1699
<code>columns</code>	<u>777</u>
<code>columns*</code>	<u>1527</u>
<code>columns-sep</code>	<u>777</u>
<code>columns-sep*</code>	<u>1527</u>
<code>\columnsep</code>	73, 77
<code>\columnsep</code>	2447, 2590
<code>\columnseprule</code>	73, 77
<code>\columnseprule</code>	2451, 2592
Commands provide by enumext :	
<code>\anskey</code> 23, 24, 50–52, 55, 57, 59–61, 63, 72, 85, 96, 97, 100	
<code>\anspic*</code>	23, 61–63, 78–80, 96, 97
<code>\anspic</code>	55, 78–80, 100
<code>\getkeyans</code>	55, 96, 101
<code>\item*</code>	23, 55, 61–63, 65, 66, 86, 93, 96, 97
<code>\itemwidth</code>	81, 89
<code>\item</code>	65, 66, 81, 85–87, 89, 92–94
<code>\miniright</code>	23, 39, 46, 47, 73, 74, 76, 77, 101
<code>\printkeyans</code>	24, 55, 96
<code>\setenumext</code>	23, 97–99
Counters defined by enumext :	
<code>enumXiii</code>	22, 29
<code>enumXii</code>	22, 29
<code>enumXiv</code>	22, 29
<code>enumXi</code>	22, 29

enumXviii 22, 29, 94
enumXvii 22, 29, 87
enumXvi 22, 29
enumXv 22, 29

cs commands:

\cs_generate_variant:Nn 317, 333, 532, 548, 1552,
1561, 1566, 1646, 2225, 2729
\cs_if_exist:NTF 287
\cs_new:Nn 188
\cs_new:Npn 192, 197, 207
\cs_new_eq:NN 223, 224, 225, 229, 230, 262, 263, 266,
267
\cs_new_protected:Nn . 234, 398, 418, 450, 713, 717,
721, 725, 729, 733, 737, 741, 745, 749, 753, 757, 761,
765, 769, 773, 809, 821, 845, 862, 873, 897, 972, 996,
1013, 1075, 1092, 1114, 1149, 1155, 1230, 1244, 1258,
1269, 1280, 1291, 1302, 1313, 1361, 1372, 1433, 1445,
1462, 1567, 1591, 1598, 1626, 1633, 1750, 1877, 1892,
1920, 1946, 2028, 2032, 2051, 2104, 2138, 2154, 2164,
2180, 2330, 2383, 2401, 2408, 2431, 2461, 2480, 2535,
2558, 2576, 2601, 2612, 2649, 2692, 2705, 2725, 2730,
2746, 2814, 2833, 2884, 2941, 2947, 2967, 2977, 2994,
3144, 3170, 3238, 3257, 3306, 3327, 3333, 3346, 3398,
3508
\cs_new_protected:Npn 180, 184, 270, 285, 302, 312,
318, 406, 425, 519, 533, 1177, 1196, 1340, 1401, 1553,
1562, 1682, 1827, 1839, 1861, 1930, 1968, 1976, 2061,
2080, 2115, 2127, 2194, 2228, 2274, 2345, 2354, 2554,
2700, 2765, 2901, 2912, 3000, 3007, 3023, 3031, 3036,
3048, 3189, 3320, 3352, 3359, 3375, 3383, 3393, 3527,
3540, 3587, 3665, 3677
\cs_new_protected_nopar:Nn . . . 2987, 3120, 3339,
3484
\cs_new_protected_nopar:Npn 3054, 3441
\cs_set:Nn 195, 1832
\cs_set:Npn 1760, 1798, 3533
\cs_set_eq:NN 194, 199, 2874, 2875, 3056, 3297, 3298,
3443
\cs_set_protected:Nn 213, 637, 653, 665, 677
\cs_set_protected:Npn . 30, 44, 52, 64, 70, 96, 125,
136, 148, 155, 215, 334, 356, 385, 466, 486, 549, 569,
613, 632, 689, 698, 777, 794, 1213, 1383, 1480, 1509,
1527, 1752, 1881, 2012, 2226, 2272
\cs_to_str:N 304, 327

D

\d 203
\DeclareDocumentEnvironment 890

dim commands:

\dim_abs:n 2199, 2204
\dim_add:Nn 2710, 3415, 3436
\dim_compare:nNnTF . 639, 655, 667, 679, 1179, 1198,
2196, 2201, 2207, 2213, 2215, 2217, 2413, 2436, 2562,
2580, 2702, 2748, 2816, 3172, 3240
\dim_compare:nTF 1723
\dim_gset_eq:NN 2825, 3249
\dim_gzero:N 2859, 3283
\dim_new:N 40, 47, 48, 49, 66, 90, 103, 113, 164, 165, 171
\dim_set:Nn . . 315, 627, 1539, 2094, 2199, 2204, 2206,
2209, 2210, 2214, 2216, 2219, 2220, 2222, 2416, 2439,
2564, 2582, 2732, 2750, 2757, 2800, 2818, 3050, 3174,
3181, 3224, 3242
\dim_set_eq:NN 473, 493, 509, 513, 2089, 2237, 2283,
2373, 2447, 2590, 2807, 2810, 2811, 2931, 3041, 3231,
3234, 3235

\dim_use:N 640, 648, 1180, 1186, 1636, 1639, 1644, 2159,
2161, 2414, 2419, 2420, 2427, 2437, 2441, 2442, 2444
\dim_zero:N 2451, 2592, 2711, 2712, 2713
\dim_zero_new:N 2763, 3187
\c_zero_dim 642, 656, 668, 680, 1180, 1198, 1725, 2196,
2201, 2207, 2214, 2414, 2437, 2562, 2580, 2748, 2816,
3172, 3240

E

\end . . 1183, 1201, 1595, 1630, 2466, 2490, 2605, 2622, 2837,
2853, 3261, 3277, 3598, 3603
\endlist 27
\endlist 224
\endlrbox 3123, 3487
\endminipage 27
\endminipage 230
enumext 4, 2311

enumext internal commands:

__enumext_add_pre_parsep: . . . 40, 819, 821, 821
__enumext_after_args_exec: . 37, 713, 725, 2323
__enumext_after_args_exec_v: 38, 729, 741, 2528
__enumext_after_args_exec_vii: . . . 745, 769
__enumext_after_args_exec_viii: 773
__enumext_after_env:n 74
__enumext_after_env:nn . . 75, 89, 184, 184, 2505,
2842, 3158, 3266
__enumext_after_hyperref: . . . 27, 232, 234, 234
__enumext_after_list: 74, 85, 92, 2328, 2480, 2480
\l__enumext_after_list_args_v_tl 743
\l__enumext_after_list_args_vii_tl 771, 3114
\l__enumext_after_list_args_viii_tl 775, 3478
__enumext_after_list_v: . . 77, 2533, 2612, 2612
__enumext_after_list_vii: . . . 2882, 2947, 2947
__enumext_after_list_viii: . . 3304, 3333, 3333
__enumext_after_star_env:nn 82
__enumext_after_stop_list: . . . 37, 38, 713, 721,
2494
__enumext_after_stop_list_v: 38, 729, 737, 2627
\l__enumext_after_stop_list_v_tl 739
__enumext_after_stop_list_vii: 745, 761, 2950
\l__enumext_after_stop_list_vii_tl . . . 763
__enumext_after_stop_list_viii: . 765, 3336
\l__enumext_after_stop_list_viii_tl . . . 767
\l__enumext_align_label_vii_str . . 3106, 3110
\l__enumext_align_label_viii_str . 3470, 3474
\l__enumext_align_label_X_str 155
\c__enumext_all_envs_clist . . 175, 355, 568, 631,
697, 712, 793, 1229
\c__enumext_all_families_seq . . 98, 3627, 3636,
3658
__enumext_anskey_wrapper:n 1484, 1837
__enumext_at_begin_document:n . . 27, 180, 180,
221, 227
__enumext_before_args_exec: 37, 713, 713, 2411
__enumext_before_args_exec_v: 37, 38, 729, 729,
2561
__enumext_before_args_exec_vii: . . 745, 745,
2944
__enumext_before_args_exec_viii: 749, 3330
__enumext_before_keys_exec: 37, 713, 717, 2321
__enumext_before_keys_exec_v: . . 38, 729, 733,
2526
__enumext_before_keys_exec_vii 745
__enumext_before_keys_exec_vii: 38, 753, 2870

`__enumext_before_keys_exec_viii:` .. 38, 757, 3293
`__enumext_before_list:` ... 73, 2315, 2408, 2408
`__enumext_before_list_v:` . 76, 2521, 2558, 2558
`__enumext_before_list_vii:` 84, 2865, 2941, 2941
`__enumext_before_list_viii:` .. 92, 3289, 3327, 3327
`\l__enumext_before_no_starred_key_v_tl` 735
`\l__enumext_before_no_starred_key_vii_-tl` 755
`\l__enumext_before_no_starred_key_viii_-tl` 759
`\l__enumext_before_starred_key_v_tl` ... 731
`\l__enumext_before_starred_key_vii_tl` . 747
`\l__enumext_before_starred_key_viii_tl` 751
`__enumext_calc_hspace:NNNNNN` 68, 2194, 2194, 2225, 2230, 2276
`__enumext_check_ans_active:` .. 53, 1445, 1445, 2511
`__enumext_check_ans_active_vii:` . 1445, 1462, 3164
`\l__enumext_check_ans_bool` ... 50, 65, 117, 1356, 1387, 1391, 1435, 1674, 1963, 2065, 2096, 2473, 2959, 3068
`__enumext_check_ans_count:` . 53, 73, 1433, 1433, 2412
`__enumext_check_ans_int:n` .. 50, 52, 1358, 1401, 1401
`\g__enumext_check_ans_item_tl` .. 63, 117, 1962, 1970, 1974
`\g__enumext_check_ans_show_bool` 74, 117, 2476, 2507, 2514
`\g__enumext_check_ans_show_h_bool` 117, 2961, 3160, 3167
`\l__enumext_columns_sep_v_dim` 2580, 2582, 2590
`\l__enumext_columns_sep_vii_dim` .. 2748, 2750, 2759, 2804, 2933, 3142
`\l__enumext_columns_sep_viii_dim` . 3172, 3174, 3183, 3228, 3506
`\l__enumext_columns_v_int` 1018, 2578, 2586, 2598, 2603
`\l__enumext_columns_vii_int` .. 2753, 2756, 2760, 2768, 2772, 2775, 2781, 2787, 2791, 2920, 3137, 3148
`\l__enumext_columns_viii_int` . 3177, 3180, 3184, 3192, 3196, 3199, 3205, 3211, 3215, 3501, 3512
`\l__enumext_compare_items_ans_int` 117, 1447, 1453, 1464, 1471
`\g__enumext_count_item_all_int` 117, 1421, 1424, 1449, 1466, 2067, 2098, 3077
`\g__enumext_count_item_i_int` 1426, 1466
`\g__enumext_count_item_ii_int` 1427, 1449, 1467
`\g__enumext_count_item_iii_int` 1428, 1450, 1467
`\g__enumext_count_item_iv_int` 1429, 1450, 1468
`\g__enumext_count_item_vii_int` 1430
`\g__enumext_count_item_with_ans_int` 52, 57, 63, 117, 1431, 1453, 1471, 1676, 1965
`\g__enumext_count_item_X_int` 117
`\g__enumext_count_level_vii_int` 3078
`\g__enumext_count_level_X_int` 117
`\l__enumext_counter_i_tl` 30, 294
`\l__enumext_counter_ii_tl` 30, 295
`\l__enumext_counter_iii_tl` 30, 296
`\l__enumext_counter_iv_tl` 30, 297
`\l__enumext_counter_style_for_ref_vii_-tl` 433, 443, 454, 456
`\l__enumext_counter_style_for_ref_viii_-tl` 460, 462
`\l__enumext_counter_style_for_ref_X_tl` 144
`\c__enumext_counter_style_tl` ... 31, 144, 400
`\g__enumext_counter_styles_tl` . 22, 29, 40, 305, 323
`\l__enumext_counter_v_tl` 30, 298
`\l__enumext_counter_vi_tl` 30, 299
`\l__enumext_counter_vii_tl` 30, 300, 430
`\l__enumext_counter_viii_tl` 30, 301, 440
`\l__enumext_current_widest_dim` 22, 40, 329, 474, 494, 510, 514
`__enumext_default_item:n` ... 2061, 2061, 2112
`__enumext_define_counters:Nn` 22, 285, 285, 294, 295, 296, 297, 298, 299, 300, 301
`__enumext_endminipage:` . 27, 227, 230, 896, 2742, 3122, 3486
`__enumext_fake_item:` 637, 637, 2263
`\l__enumext_fake_item_indent_v_dim` 656, 661
`\l__enumext_fake_item_indent_v_tl` 658, 2120, 2124, 2132
`\l__enumext_fake_item_indent_vii_dim` 668, 673
`\l__enumext_fake_item_indent_vii_tl` 670, 3118
`\l__enumext_fake_item_indent_viii_dim` . 680, 685, 3415, 3436
`\l__enumext_fake_item_indent_viii_tl` .. 682, 3482
`\l__enumext_fake_item_indent_X_tl` 70
`__enumext_fake_item_vii:` 637, 665, 2299
`__enumext_fake_item_viii:` 637, 677, 2304
`\g__enumext_footnote_arg_seq` . 141, 2034, 2047, 2057
`\g__enumext_footnote_int` . 141, 2041, 2044, 2046, 2048
`\g__enumext_footnote_int_seq` . 141, 2035, 2048, 2053, 2056
`__enumext_footnotes_key_bool` 27
`\l__enumext_footnotes_key_bool` 24, 27, 87, 131, 243, 248, 257, 3085, 3133, 3458, 3497
`__enumext_footnotetext:nn` ... 2028, 2028, 2058
`__enumext_getkeyans:nn` ... 96, 3536, 3540, 3540
`__enumext_getkeyans_aux:n` . 96, 3524, 3527, 3527
`\l__enumext_hyperref_bool` . 24, 27, 28, 131, 239, 260, 277, 1740, 1950, 3062, 3449
`__enumext_hypertarget:nn` 28, 234, 262, 266, 282
`__enumext_if_is_int:n` 201
`__enumext_if_is_int:nTF` 201, 521, 535
`\l__enumext_item_column_pos_vii_int` 85, 2775, 2781, 2787, 2791, 2798, 2990, 3137, 3140
`\l__enumext_item_column_pos_viii_int` ... 92, 3199, 3205, 3211, 3215, 3222, 3342, 3501, 3504
`\l__enumext_item_column_pos_X_int` 155
`\g__enumext_item_count_all_vii_int` 85, 2799, 2991, 3148, 3155
`\g__enumext_item_count_all_viii_int` 92, 3223, 3343, 3512, 3519
`\g__enumext_item_count_all_X_int` 155
`__enumext_item_peek_args_vii:` .. 85, 86, 2992, 2994, 2994
`__enumext_item_peek_args_viii:` 92, 3344, 3346, 3346
`__enumext_item_starred:` .. 67, 2154, 2154, 2172
`\l__enumext_item_starred_vii_bool` 3009, 3025, 3089


```

\l__enumext_item_starred_viii_bool 3361, 3377,
    3462
\l__enumext_item_starred_X_bool ..... 155
\__enumext_item_std:w 27, 65, 67, 80, 221, 225, 2071,
    2077, 2102, 2120, 2124, 2132, 2723
\g__enumext_item_symbol_aux_vii_tl 3033, 3091,
    3094, 3098, 3100
\g__enumext_item_symbol_aux_X_tl ..... 155
\l__enumext_item_symbol_sep_vii_dim .. 3042,
    3050, 3097, 3099
\g__enumext_item_symbol_tl 22, 65, 35, 2086, 2160,
    2177
\l__enumext_item_symbol_vii_tl ..... 3094
\l__enumext_item_text_vii_box 3084, 3125, 3132
\l__enumext_item_text_viii_box 3457, 3489, 3496
\l__enumext_item_text_X_box ..... 155
\l__enumext_item_width_vii_dim ... 2757, 2802,
    2810, 2811
\l__enumext_item_width_viii_dim .. 3181, 3226,
    3234, 3235
\l__enumext_item_width_X_dim ..... 155
\l__enumext_itemindent_X_dim ..... 44
\l__enumext_itemsep_vii_skip ..... 3154
\l__enumext_itemsep_viii_skip ..... 3518
\l__enumext_joined_item_aux_vii_int .. 2796,
    2797, 2798, 2799, 2805
\l__enumext_joined_item_aux_viii_int . 3220,
    3221, 3222, 3223, 3229
\l__enumext_joined_item_aux_X_int .... 155
\__enumext_joined_item_vii:w .. 86, 2997, 2998,
    3000, 3000
\l__enumext_joined_item_vii_int .. 2767, 2768,
    2771, 2773, 2779, 2784, 2789, 2794, 2796, 2802
\__enumext_joined_item_viii:w . 92, 3349, 3350,
    3352, 3352
\l__enumext_joined_item_viii_int . 3191, 3192,
    3195, 3197, 3203, 3208, 3213, 3218, 3220, 3226
\l__enumext_joined_item_X_int ..... 155
\l__enumext_joined_width_vii_dim . 2800, 2807,
    2810, 3115, 3127
\l__enumext_joined_width_viii_dim 3224, 3231,
    3234, 3479, 3491
\l__enumext_joined_width_X_dim ..... 155
\__enumext_keyans_addto_prop:n 61, 1861, 1861,
    2134, 2655
\__enumext_keyans_addto_seq:n . 62, 1930, 1930,
    2136, 2657
\__enumext_keyans_addto_seq_link: 1930, 1944,
    1946, 3408
\__enumext_keyans_anspic_code:nnn . 78, 2646,
    2649, 2649
\__enumext_keyans_check_ans:nn 63, 1968, 1968,
    2531, 2687
\__enumext_keyans_default_item:n .. 66, 2115,
    2115, 2150
\l__enumext_keyans_env_bool 20, 2387, 2542, 2626
\__enumext_keyans_fake_item: .. 637, 653, 2253
\l__enumext_keyans_level_h_int 20, 1908, 3308,
    3309
\l__enumext_keyans_level_int .. 20, 1171, 1663,
    1903, 2541, 2545, 2640
\__enumext_keyans_make_label: 30, 68, 2180, 2180,
    2252
\__enumext_keyans_mini_addvspace: 45, 76, 1075,
    1075, 2570
\__enumext_keyans_mini_right_cmd:n 47, 1173,
    1196, 1196
\__enumext_keyans_mini_set_vskip: . 44, 1013,
    1013, 1077
\__enumext_keyans_multi_addvspace: . 77, 862,
    873, 2595
\__enumext_keyans_multi_set_vskip: . 41, 862,
    862, 875
\__enumext_keyans_multicols_start: 76, 2574,
    2576, 2576
\__enumext_keyans_multicols_stop: . 77, 1200,
    2601, 2601, 2625
\__enumext_keyans_parse_keys:n 2520, 2554, 2554
\l__enumext_keyans_pic_above_int . 112, 2733,
    2734, 2736
\l__enumext_keyans_pic_above_skip .. 80, 112,
    2678, 2717
\__enumext_keyans_pic_arg_two: 80, 2676, 2705,
    2705
\l__enumext_keyans_pic_below_int . 112, 2733,
    2734, 2737
\l__enumext_keyans_pic_body_seq .. 78-80, 112,
    2644, 2683, 2741
\__enumext_keyans_pic_do:n 80, 2683, 2685, 2725,
    2725, 2729
\l__enumext_keyans_pic_level_int .. 20, 1163,
    1667, 1864, 1898, 1933, 2694, 2695
\__enumext_keyans_pic_row:n 80, 2727, 2730, 2730
\__enumext_keyans_pic_safe_exec: .. 79, 2672,
    2692, 2692
\__enumext_keyans_pic_skip_abs:N .. 80, 2700,
    2700, 2716
\l__enumext_keyans_pic_width_dim . 112, 2732,
    2739
\__enumext_keyans_redefine_item: .. 67, 2138,
    2138, 2251
\__enumext_keyans_safe_exec: . 2519, 2535, 2535
\__enumext_keyans_show_left:n . 67, 1976, 1976,
    2130, 2663
\__enumext_keyans_starred_item:n .. 67, 2127,
    2127, 2146
\__enumext_keyans_store_ref: .. 61, 1877, 1877,
    2135, 2656, 3406
\__enumext_keyans_store_ref_aux_i: 62, 1877,
    1889, 1892
\__enumext_keyans_store_ref_aux_ii: 62, 1877,
    1918, 1920
\l__enumext_keyans_tmpa_dim 82, 3414, 3415, 3435,
    3436
\l__enumext_keyans_tmpa_tl .. 23, 82, 2129, 2133,
    3388, 3411, 3413, 3414, 3432, 3434, 3435
\l__enumext_keyans_tmpb_tl 82, 3389, 3401, 3403
\l__enumext_label_copy_i_tl .. 1794, 1896, 1901,
    1906, 1911
\l__enumext_label_copy_v_tl ..... 1906
\l__enumext_label_copy_vi_tl ..... 1901
\l__enumext_label_copy_vii_tl 1769, 1780, 1811,
    1896
\l__enumext_label_copy_viii_tl ..... 1911
\l__enumext_label_copy_X_tl ..... 133
\l__enumext_label_fill_left_v_tl ..... 2184
\l__enumext_label_fill_left_X_tl ..... 70
\l__enumext_label_fill_right_v_tl .... 2191
\l__enumext_label_fill_right_X_tl ..... 70

```

```

\l__enumext_label_font_style_v_tl 2185, 2667
\l__enumext_label_font_style_vii_tl ... 3103
\l__enumext_label_font_style_viii_tl .. 3467
\l__enumext_label_i_tl ..... 466
\l__enumext_label_ii_tl ..... 466
\l__enumext_label_iii_tl ..... 466
\l__enumext_label_iv_tl ..... 466
\__enumext_label_style:Nnn 22, 29, 318, 318, 333,
471, 491, 507, 511
\l__enumext_label_v_tl .. 61, 62, 504, 1869, 1938,
1980, 1987, 2002, 2009, 2129, 2133, 2523, 2662, 2664
\l__enumext_label_vi_tl . 61, 62, 504, 1866, 1935,
2662, 2664, 2668
\l__enumext_label_vii_tl . 486, 3020, 3045, 3052
\l__enumext_label_viii_tl 486, 3372, 3396, 3400,
3413, 3434
\l__enumext_label_width_by_box .. 40, 314, 315
\__enumext_label_width_by_box:Nn 29, 312, 312,
317, 329, 545, 3414, 3435
\l__enumext_labelsep_i_dim ... 1984, 2006, 3417,
3438
\l__enumext_labelsep_v_dim ..... 2585
\l__enumext_labelsep_vii_dim . 2752, 2761, 2803,
3043, 3113, 3129
\l__enumext_labelsep_viii_dim 3176, 3185, 3227,
3477, 3493
\l__enumext_labelwidth_i_dim . 1983, 2005, 3417,
3438
\l__enumext_labelwidth_v_dim ..... 2585
\l__enumext_labelwidth_vii_dim ... 2752, 2760,
2803, 3106, 3110, 3128
\l__enumext_labelwidth_viii_dim .. 3176, 3184,
3227, 3470, 3474, 3492
\l__enumext_leftmargin_tmp_v_bool . 80, 2707
\l__enumext_leftmargin_tmp_X_bool ..... 44
\l__enumext_leftmargin_tmp_X_dim ..... 44
\l__enumext_leftmargin_X_dim ..... 44
\__enumext_level: 188, 188, 194, 195, 199, 409, 411,
412, 420, 422, 640, 644, 648, 715, 719, 723, 727, 811,
813, 815, 817, 850, 852, 854, 856, 860, 900, 903, 922,
931, 937, 942, 946, 957, 961, 962, 967, 1003, 1007,
1180, 1186, 1233, 1235, 1237, 1240, 1247, 1249, 1251,
1254, 1439, 1440, 1442, 1571, 1579, 1583, 1587, 1832,
1835, 1836, 2068, 2070, 2071, 2075, 2076, 2077, 2084,
2086, 2090, 2091, 2094, 2099, 2101, 2102, 2156, 2159,
2161, 2168, 2169, 2170, 2173, 2176, 2318, 2320, 2356,
2361, 2362, 2363, 2365, 2369, 2374, 2375, 2376, 2378,
2391, 2396, 2403, 2414, 2416, 2419, 2420, 2422, 2427,
2434, 2437, 2439, 2441, 2442, 2443, 2444, 2447, 2453,
2458, 2464, 2467, 2469, 2482, 3073, 3074
\__enumext_level_ ..... 194
\__enumext_level_end:n ..... 192, 197
\l__enumext_level_h_int 20, 428, 452, 1788, 1805,
2339, 2389, 2886, 2887
\l__enumext_level_int 20, 190, 823, 974, 1167, 1418,
1765, 1775, 1781, 1787, 1795, 1803, 1810, 2266, 2332,
2333, 2348, 2394, 2449, 2509, 2549, 2636, 2895, 2971,
2981, 3162, 3315
\__enumext_level_set:n ..... 192, 192
\__enumext_list_arg_two_i: ..... 2226
\__enumext_list_arg_two_ii: ..... 2226
\__enumext_list_arg_two_iii: ..... 2226
\__enumext_list_arg_two_iv: ..... 2226
\__enumext_list_arg_two_v: . 67, 2226, 2525, 2708
\__enumext_list_arg_two_vii: ..... 2272, 2869
\__enumext_list_arg_two_viii: .... 2272, 3292
\l__enumext_listoffset_v_dim ..... 2587
\l__enumext_listparindent_vii_dim .... 3116
\l__enumext_listparindent_viii_dim ... 3480
\__enumext_make_label: 30, 65, 66, 68, 2164, 2164,
2261
\l__enumext_mark_answer_sym_tl . 56, 107, 1487,
1641, 1847, 1991, 3421
\l__enumext_mark_position_str 107, 1491, 1492,
1514, 1515, 1639
\l__enumext_mark_ref_sym_tl .. 107, 1501, 1745,
1958
\__enumext_mini_addvspace: 43, 73, 996, 996, 2424
\__enumext_mini_addvspace_vii: 46, 1149, 1149,
2828
\__enumext_mini_addvspace_viii: 46, 1149, 1155,
3252
__enumext_mini_env* ..... 890
\__enumext_mini_right_cmd:n 47, 1175, 1177, 1177
\__enumext_mini_set_vskip: ... 42, 897, 897, 998
\__enumext_mini_set_vskip_vii: 45, 1092, 1092,
1151
\__enumext_mini_set_vskip_viii: 45, 1092, 1114,
1157
\__enumext_minipage:w 27, 227, 229, 892, 2739, 3115,
3479
\l__enumext_minipage_active_v_bool ... 76, 77,
2568, 2593, 2606, 2614
\g__enumext_minipage_active_vii_bool ... 82,
2839, 2844, 2856
\l__enumext_minipage_active_vii_bool . 2824,
2835
\g__enumext_minipage_active_viii_bool 3263,
3268, 3280
\l__enumext_minipage_active_viii_bool 3248,
3259
\g__enumext_minipage_active_X_bool ... 155
\l__enumext_minipage_active_X_bool .... 58
\g__enumext_minipage_after_skip 58, 1096, 1108,
2854, 3278
\l__enumext_minipage_after_skip 42, 43, 74, 77,
58, 913, 928, 948, 964, 979, 985, 991, 1005, 1015, 1024,
1027, 1039, 1057, 1068, 1084, 1116, 1129, 1143, 2491,
2623
\g__enumext_minipage_center_vii_bool . 2848,
2857
\g__enumext_minipage_center_viii_bool 3272,
3281
\g__enumext_minipage_center_X_bool ... 155
\l__enumext_minipage_hsep_v_dim ... 76, 2566
\l__enumext_minipage_hsep_vii_dim .... 2822
\l__enumext_minipage_hsep_viii_dim ... 3246
\l__enumext_minipage_left_skip 42, 76, 58, 905,
920, 939, 954, 1001, 1011, 1016, 1022, 1031, 1048,
1060, 1080, 1090, 1094, 1099, 1103, 1117, 1121, 1135,
1153, 1159
\l__enumext_minipage_left_v_dim 76, 2564, 2572
\l__enumext_minipage_left_vii_dim 2818, 2830
\l__enumext_minipage_left_viii_dim 3242, 3254
\l__enumext_minipage_left_X_dim ..... 58
\g__enumext_minipage_right_skip 58, 1095, 1100,
1104, 2847, 3271
\l__enumext_minipage_right_skip .. 42, 58, 909,
924, 944, 959, 1017, 1023, 1035, 1053, 1064, 1118,

```

1125, 1139, 1187, 1204
 \l__enumext_minipage_right_v_dim .. 76, 1198,
 1203, 2562, 2566
 \g__enumext_minipage_right_vii_dim 82, 2826,
 2846, 2859
 \l__enumext_minipage_right_vii_dim 82, 2816,
 2821, 2827
 \g__enumext_minipage_right_viii_dim .. 3250,
 3270, 3283
 \l__enumext_minipage_right_viii_dim .. 3240,
 3245, 3251
 \g__enumext_minipage_right_X_dim 155
 \g__enumext_minipage_right_X_skip 155
 \g__enumext_minipage_stat_int . 73, 76, 58, 1192,
 1209, 2423, 2484, 2489, 2569, 2616, 2621
 \g__enumext_miniright_code_vii_tl . 83, 2852,
 2858
 \g__enumext_miniright_code_viii_tl 3276, 3282
 \g__enumext_miniright_code_X_tl 155
 __enumext_multi_addvspace: ... 40, 74, 845, 845,
 2455
 __enumext_multi_set_vskip: .. 40, 809, 809, 847
 \l__enumext_multicols_above_ii_skip ... 828
 \l__enumext_multicols_above_iii_skip .. 834
 \l__enumext_multicols_above_iv_skip ... 840
 \l__enumext_multicols_above_v_skip 864, 878,
 888
 \l__enumext_multicols_above_X_skip 52
 \l__enumext_multicols_below_v_skip 868, 882,
 2608
 \l__enumext_multicols_below_X_skip 52
 __enumext_multicols_start: 73, 2429, 2431, 2431
 __enumext_multicols_stop: 74, 1182, 2461, 2461,
 2493
 __enumext_newlabel:nn 24, 28, 60, 270, 270, 1821,
 1924
 \l__enumext_newlabel_arg_one_tl 24, 28, 60, 62,
 133, 1744, 1814, 1822, 1913, 1925, 1956
 \l__enumext_newlabel_arg_two_tl 24, 28, 59, 133,
 1768, 1778, 1792, 1808, 1823, 1900, 1905, 1910, 1926
 __enumext_parse_keys:n 2314, 2345, 2345
 __enumext_parse_keys_vii:n .. 2864, 2901, 2901
 __enumext_parse_keys_viii:n . 3288, 3320, 3320
 __enumext_parse_store_keys:n 71, 72, 2351, 2354,
 2354
 __enumext_parse_store_keys_vii:n . 84, 2908,
 2912, 2912
 \l__enumext_parsep_i_skip . 826, 828, 977, 1025
 \l__enumext_parsep_ii_skip 832, 834, 983
 \l__enumext_parsep_iii_skip 838, 840, 989
 \l__enumext_parsep_vii_skip 3117
 \l__enumext_parsep_viii_skip 3481
 \l__enumext_partopsep_v_skip .. 880, 884, 1051,
 1055, 1062, 1066, 1082, 1086
 \l__enumext_partopsep_viii_skip 1127
 __enumext_phantomsection: 28, 234, 263, 267, 283
 __enumext_print_footnote: ... 2028, 2051, 3135,
 3499
 __enumext_print_keyans_box:NN 56, 1633, 1633,
 1646, 1834, 1982, 2004, 3417, 3438
 \l__enumext_print_keyans_i_tl 3550, 3579
 \l__enumext_print_keyans_ii_tl ... 3555, 3580
 \l__enumext_print_keyans_iii_tl .. 3560, 3581
 \l__enumext_print_keyans_iv_tl ... 3565, 3582
 \l__enumext_print_keyans_vii_tl .. 3570, 3583
 \l__enumext_print_keyans_X_tl 96
 __enumext_printkeyans:nnn . 97, 3584, 3587, 3587
 __enumext_redefine_item: . 66, 2104, 2104, 2260
 \l__enumext_ref_aux_tl 144, 409, 411, 414, 430, 432,
 435, 440, 442, 445
 \l__enumext_ref_key_arg_tl .. 144, 403, 408, 415,
 427, 436, 446
 __enumext_regex_label_ref_key: . 31, 398, 398,
 410, 431, 441
 __enumext_register_counter_style:Nn .. 302,
 302, 307, 308, 309, 310, 311
 __enumext_remove_extra_parsep_vii: .. 2879,
 3144, 3144
 __enumext_remove_extra_parsep_viii: . 3302,
 3508, 3508
 __enumext_renew_footnote: ... 2028, 2032, 3087,
 3460
 \l__enumext_resume_bool 22, 35, 1370, 2243
 __enumext_resume_counter: . 51, 1328, 1361, 1361
 __enumext_resume_counter_star: 1330
 __enumext_resume_counter_vii: 51, 1337, 1361,
 1372
 \g__enumext_resume_int 22, 75, 35, 1365, 1376, 2244,
 2497
 \l__enumext_resume_vii_bool ... 35, 1381, 2290
 \g__enumext_resume_vii_int .. 85, 35, 2291, 2952
 __enumext_safe_exec: 2313, 2330, 2330
 __enumext_safe_exec_vii: ... 2863, 2884, 2884
 __enumext_safe_exec_viii: ... 3287, 3306, 3306
 __enumext_set_error:nn 3665, 3675, 3677
 __enumext_set_label_ref:n ... 31, 406, 406, 478
 __enumext_set_label_ref_h:n . 31, 425, 425, 498
 __enumext_set_parse:n 3648, 3665, 3665
 \l__enumext_setkey_tmpa_int ... 91, 3641, 3645
 \l__enumext_setkey_tmpa_seq 91, 3639, 3649, 3655,
 3657, 3659, 3672
 \l__enumext_setkey_tmpa_tl 91, 3647, 3651
 \l__enumext_setkey_tmpb_seq 91, 3640, 3643, 3647,
 3648
 \l__enumext_setkey_tmpb_tl 91, 3667, 3669, 3670
 \l__enumext_show_answer_bool . 107, 1495, 1499,
 1518, 1522, 1841, 1978, 2659, 3409
 __enumext_show_length:nnn .. 36, 207, 207, 3717,
 3718, 3719, 3720, 3721, 3722, 3723, 3724, 3725, 3726,
 3732, 3733, 3734, 3735, 3736, 3737, 3738, 3739, 3740,
 3741
 \l__enumext_show_position_bool . 63, 107, 1496,
 1498, 1519, 1521, 1845, 1989, 2660, 3419
 \g__enumext_standar_bool 20, 2342
 \l__enumext_standar_bool . 20, 1773, 1786, 1802,
 2335, 2496, 2894
 __enumext_standard_item_vii:w 86, 3005, 3007,
 3007
 __enumext_standard_item_viii:w . 92, 93, 3357,
 3359, 3359
 \g__enumext_starred_bool . 83, 85, 20, 1417, 1764,
 1774, 1804, 1894, 2474, 2898, 2959, 2964
 \l__enumext_starred_bool . 83, 85, 20, 1697, 1705,
 1789, 1830, 2338, 2891, 2965
 __enumext_starred_columns_set_vii: .. 2746,
 2746, 2872
 __enumext_starred_columns_set_viii: . 3170,
 3170, 3295

```

\__enumext_starred_item:nn ... 2080, 2080, 2110
\__enumext_starred_item_exec: ... 3398, 3464
\__enumext_starred_item_exec_viii: 93, 3398
\__enumext_starred_item_vii:w . 86, 3004, 3023,
    3023
\__enumext_starred_item_vii_aux_i:w . 3023,
    3028, 3031
\__enumext_starred_item_vii_aux_ii:w . 3023,
    3029, 3034, 3036
\__enumext_starred_item_vii_aux_iii:w 3023,
    3039, 3048
\__enumext_starred_item_viii:w . 92, 93, 3356,
    3375, 3375
\__enumext_starred_item_viii_aux_i:w . 3375,
    3380, 3383
\__enumext_starred_item_viii_aux_ii:w 3375,
    3381, 3391, 3393
\__enumext_starred_joined_item_vii:n . 81, 86,
    2765, 2765, 3002
\__enumext_starred_joined_item_viii:n 89, 92,
    3189, 3189, 3354
\__enumext_start_from:NNn 33, 519, 519, 532, 554
\__enumext_start_item_tmp_vii: 83, 2875, 2987,
    2987
\__enumext_start_item_tmp_viii: 91, 3298, 3339,
    3339
\__enumext_start_item_vii:w . 86, 87, 3015, 3020,
    3045, 3052, 3054, 3054
\__enumext_start_item_viii:w 93, 94, 3367, 3372,
    3396, 3441, 3441
\__enumext_start_list:nn 27, 69, 80, 221, 223, 2317,
    2522, 2673, 2867, 3290
\__enumext_start_mini_vii: . 84, 2814, 2814, 2945
\__enumext_start_mini_viii: 92, 3238, 3238, 3331
\__enumext_start_store_level: . 72, 2316, 2383,
    2383
\__enumext_start_store_level_vii: . 85, 2866,
    2967, 2967
\l__enumext_start_X_int . . . . . 70, 549
\__enumext_stop_item_tmp_vii: . 83, 85, 87, 2874,
    2878, 2989, 3056
\__enumext_stop_item_tmp_viii: 91, 92, 94, 3297,
    3301, 3341, 3443
\__enumext_stop_item_vii: 87, 88, 3056, 3120, 3120
\__enumext_stop_item_viii: . 94, 95, 3443, 3484,
    3484
\__enumext_stop_list: . 27, 221, 224, 2326, 2532,
    2686, 2880, 3303
\__enumext_stop_mini_vii: 82, 85, 2833, 2833, 2949
\__enumext_stop_mini_viii: . 92, 3238, 3257, 3335
\__enumext_stop_store_level: . 72, 2327, 2383,
    2401
\__enumext_stop_store_level_vii: . 85, 2881,
    2967, 2977
\l__enumext_store_active_bool 23, 50, 71, 84, 82,
    1351, 1363, 1374, 1659, 2349, 2386, 2537, 2544, 2632,
    2690, 2906, 2969, 2979, 3314
\__enumext_store_addto_prop:n 55, 61, 1552, 1553,
    1561, 1684, 1875, 3405
\__enumext_store_addto_seq:n 55, 63, 1562, 1562,
    1566, 1573, 1587, 1595, 1604, 1622, 1630, 1748, 1961
\l__enumext_store_ans_bool 117, 1390, 1437, 1569,
    1593, 1600, 1628, 1672, 3071
\l__enumext_store_anskey_arg_tl . 23, 58, 82,
    1690, 1699, 1701, 1707, 1715, 1718, 1728, 1733, 1736,
    1742, 1748
\__enumext_store_anskey_code:nnnn . 57, 1678,
    1682, 1682
\__enumext_store_anskey_show_left:n 60, 1689,
    1839, 1839
\__enumext_store_anskey_show_wrap:n 60, 1827,
    1827, 1843, 1858
\l__enumext_store_columns_break_bool . 1653,
    1696
\l__enumext_store_columns_join_int 82, 1704,
    1709
\l__enumext_store_columns_sep_vii_bool 2927
\l__enumext_store_columns_sep_vii_dim 2932,
    2936
\l__enumext_store_columns_sep_X_bool . . . 96
\l__enumext_store_columns_sep_X_dim . . . . 96
\l__enumext_store_columns_vii_bool . . . 2914
\l__enumext_store_columns_vii_int 2919, 2923
\l__enumext_store_columns_X_bool . . . . . 96
\l__enumext_store_columns_X_int . . . . . 96
\__enumext_store_internal_ref: . 57, 59, 1687,
    1750, 1750
\l__enumext_store_item_symbol_sep_dim 1651,
    1725, 1730
\l__enumext_store_item_symbol_tl . 1649, 1716,
    1720
\l__enumext_store_keyans_label_tl 23, 61–63,
    82, 1863, 1866, 1869, 1873, 1875, 1932, 1935, 1938,
    1942, 1952, 1961, 1962, 3385, 3400, 3403, 3405, 3407
\__enumext_store_level_close: . 55, 1567, 1591,
    2405
\__enumext_store_level_close_vii: 1598, 1626,
    2983
\__enumext_store_level_open: . 54, 55, 72, 1567,
    1567, 2392, 2397
\__enumext_store_level_open_vii: . 84, 1598,
    1598, 2973
\g__enumext_store_name_tl 23, 74, 82, 1455, 1458,
    1473, 1476, 2477, 2515, 2962, 3168
\l__enumext_store_name_tl 23, 50, 82, 1342, 1343,
    1345, 1347, 1349, 1367, 1378, 1555, 1557, 1564, 1816,
    1817, 1853, 1915, 1916, 1997, 2477, 2498, 2501, 2953,
    2956, 2962, 3427
\l__enumext_store_opt_vii_tl . 1602, 1612, 1618,
    1622, 2921, 2934
\l__enumext_store_opt_X_tl . . . . . 96
\l__enumext_store_ref_key_bool 57, 1504, 1685,
    1739, 1879, 1949
\l__enumext_store_upper_level_X_bool . . . 96
\l__enumext_store_write_aux_file_tl 24, 60, 62,
    133, 1819, 1825, 1922, 1928
\__enumext_storing_set:n 50, 51, 1326, 1335, 1340,
    1340
\l__enumext_the_counter_vii_tl . . . . . 432
\l__enumext_the_counter_viii_tl . . . . . 442
\l__enumext_the_counter_X_tl . . . . . 144
\__enumext_tmp:n 30, 34, 44, 51, 52, 57, 64, 69, 70, 81,
    96, 106, 125, 130, 136, 140, 148, 154, 155, 174, 215,
    219, 632, 636, 1383, 1400, 1480, 1508, 1509, 1526,
    1752, 1759, 1760, 1781, 1795, 1798, 1810, 1881, 1888,
    2226, 2271, 2272, 2310
\__enumext_tmp:nn 334, 355, 356, 384, 385, 397, 549,
    568, 613, 631, 689, 697, 698, 712, 777, 793, 794, 808,

```

1213, 1229, 1527, 1551, 2012, 2027

_enumext_tmp:nnn 466, 482, 483, 484, 485, 486, 502, 503

_enumext_tmp:nnnnn 569, 594, 597, 600, 602, 604, 607, 610

_enumext_tmp:w 3533, 3536

\l_enumext_tmpa_vii_int 2756, 2759

\l_enumext_tmpa_viii_int 3180, 3183

\l_enumext_tmpa_X_int 155

\l_enumext_topsep_v_skip 866, 870, 1020, 1033, 1041, 1046, 1066, 1070, 2689, 2720

\l_enumext_topsep_vii_skip .. 1097, 1106, 1110

\l_enumext_topsep_viii_skip . 1119, 1141, 1145

_enumext_use_key_ref: 31, 418, 418, 2262

_enumext_use_key_ref_h: .. 32, 450, 450, 2296

\l_enumext_vspace_a_star_v_bool 1262

\l_enumext_vspace_a_star_vii_bool ... 1284

\l_enumext_vspace_a_star_viii_bool ... 1295

\l_enumext_vspace_a_star_X_bool 70

_enumext_vspace_above: .. 48, 1230, 1230, 2410

_enumext_vspace_above_v: . 49, 1258, 1258, 2560

\l_enumext_vspace_above_v_skip .. 1260, 1264, 1266

_enumext_vspace_above_vii: .. 49, 1280, 1280, 2943

\l_enumext_vspace_above_vii_skip 1282, 1286, 1288

_enumext_vspace_above_viii: . 49, 1280, 1291, 3329

\l_enumext_vspace_above_viii_skip 1293, 1297, 1299

\l_enumext_vspace_b_star_v_bool 1273

\l_enumext_vspace_b_star_vii_bool ... 1306

\l_enumext_vspace_b_star_viii_bool ... 1317

\l_enumext_vspace_b_star_X_bool 70

_enumext_vspace_below: .. 48, 1244, 1244, 2495

_enumext_vspace_below_v: . 49, 1269, 1269, 2628

\l_enumext_vspace_below_v_skip .. 1271, 1275, 1277

_enumext_vspace_below_vii: .. 49, 1302, 1302, 2951

\l_enumext_vspace_below_vii_skip 1304, 1308, 1310

_enumext_vspace_below_viii: . 49, 1302, 1313, 3337

\l_enumext_vspace_below_viii_skip 1315, 1319, 1321

_enumext_widest_from:nnNn .. 34, 533, 533, 548, 560

\g_enumext_widest_label_tl 22, 29, 40, 322, 326, 330

\l_enumext_wrap_label_opt_v_bool 2123

\l_enumext_wrap_label_opt_vii_bool 86, 3014

\l_enumext_wrap_label_opt_viii_bool 93, 3366

\l_enumext_wrap_label_opt_X_bool 70

\l_enumext_wrap_label_v_bool 2119, 2123, 2131, 2186

\l_enumext_wrap_label_vii_bool 86, 3013, 3018, 3026, 3104

\l_enumext_wrap_label_viii_bool .. 93, 3365, 3370, 3378, 3468

\l_enumext_wrap_label_X_bool 70

_enumext_wrapper_label_v:n 2188, 2668

_enumext_wrapper_label_vii:n 3107

_enumext_wrapper_label_viii:n 3471

_enumext_zero_count_level: .. 213, 213, 1460, 1478

_enumext_zero_parsep: 43, 917, 972, 972

enumext* 4, 2861

enumXi 294

enumXii 294

enumXiii 294

enumXiv 294

enumXv 294

enumXvi 294

enumXvii 294

enumXviii 294

Environments provide by enumext:

enumext* .. 21, 22, 24–26, 29–33, 35, 36, 38, 39, 45, 46, 49–59, 62, 64, 71, 72, 83–85, 87, 89, 90, 92, 96, 99, 101

enumext 21, 22, 24, 26, 29–32, 34–37, 39–48, 50–59, 62, 64–69, 71, 72, 75, 79, 81, 82, 85, 96, 99, 100

keyans* 21–23, 25, 26, 29–33, 35, 36, 38, 39, 45, 46, 49, 51, 54, 55, 61, 64, 91, 94, 99, 101

keyanspic 21–24, 29, 30, 33, 46, 50, 51, 55, 61–63, 78–80, 100

keyans 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:

enumext* 70

keyans* 70

list 26, 27, 68, 69, 71

lrbox 81, 87, 88, 94, 95

minipage 26, 27, 39, 41, 78–81, 87, 88, 95

multicols 39–42, 47, 73, 74, 76, 77

exp commands:

\exp_after:wN 3536

\exp_args:Ne 2347, 3524

\exp_not:N 152, 325, 414, 435, 445, 646, 660, 661, 672, 673, 684, 685, 1744, 1850, 1851, 1954, 1994, 1995, 3424, 3425, 3533

\exp_not:n 414, 415, 435, 436, 445, 446, 647, 1535, 1542, 1709, 1720, 1730, 1744, 1745, 1822, 1925, 1956, 1958, 2365, 2378, 2923, 2936

F

\fbox 1485

file commands:

\file_input_stop: 3834

first 698

font 334

\footnote 64

\footnote 64, 2036

\footnotemark 2046

\footnotesize 1851, 1995, 3425

\footnotetext 2030

G

\getkeyans 13, 96, 3522

group commands:

\group_begin: .. 1671, 1849, 1993, 3083, 3102, 3423, 3456, 3466, 3544, 3578

\group_end: 1680, 1856, 2000, 3112, 3124, 3430, 3476, 3488, 3546, 3585

H

\hbadness 3131, 3495

hbox commands:

\hbox_set:Nn 314

\hfill 364, 368, 373, 374, 1184, 1202, 1744, 1954, 2838, 3262

hook commands:

 \hook_gput_code:nnn 9, 182, 186, 232

 \hook_gset_rule:nnnn 233

\hspace 3142, 3506

\hyperlink 59, 63

\hyperlink 1744, 1954

\hypertarget 28

\hypertarget 262

I

\IfHyperBoolean 240

\IfPackageLoadedTF 11, 236, 250

\ignorespaces 649

int commands:

 \int_add:Nn 2798, 3222

 \int_case:nn 823, 974

 \int_compare:nNnTF 428, 452, 899, 1018, 1163, 1167, 1171, 1452, 1470, 1663, 1667, 1864, 1898, 1903, 1908, 1933, 2333, 2389, 2394, 2433, 2449, 2463, 2484, 2509, 2545, 2549, 2578, 2603, 2616, 2636, 2640, 2695, 2768, 2778, 2794, 2887, 2971, 2981, 3137, 3146, 3162, 3192, 3202, 3218, 3309, 3315, 3501, 3510, 3645

 \int_compare_p:nNn .. 1418, 1765, 1775, 1787, 1788, 1803, 1805, 2339, 2895

 \int_decr:N 2797, 3221

 \int_eval:n 1557, 1817, 1851, 1916, 1995, 2244, 2247, 2291, 2294, 2786, 3210, 3425

 \int_from_alph:n 527, 541

 \int_from_roman:n 529, 543

 \int_gadd:Nn 1439, 2799, 3073, 3223

 \int_gincr:N 1442, 1676, 1965, 2067, 2068, 2098, 2099, 2423, 2569, 2991, 3077, 3078, 3343

 \int_gset:Nn 1365, 1376, 2044

 \int_gset_eq:NN 1421, 1424, 1426, 1427, 1428, 1429, 1430, 1431, 2041, 2497, 2500, 2952, 2955

 \int_gzero:N 217, 1192, 1209, 2489, 2621, 3155, 3519

 \int_if_exist:NTF 1352, 1403, 1405, 1407, 1409, 1411, 1413, 2498, 2953

 \int_incr:N 2332, 2541, 2694, 2886, 2990, 3308, 3342

 \int_mod:nn 3148, 3512

 \int_new:N 20, 21, 22, 23, 24, 36, 38, 58, 74, 86, 93, 101, 114, 115, 122, 123, 124, 127, 128, 141, 158, 159, 160, 161, 162, 1354, 1404, 1406, 1408, 1410, 1412, 1414

 \int_set:Nn 523, 527, 529, 1447, 1464, 1532, 1704, 2733, 2734, 2756, 2767, 2773, 2789, 3131, 3180, 3191, 3197, 3213, 3495, 3641

 \int_set_eq:NN 2360, 2796, 2918, 3220

 \int_step_function:nnN 1781, 1795, 1810

 \int_step_inline:nnn 2735, 3668

 \int_to_roman:n 190, 1761, 1799

 \int_use:N .. 900, 1367, 1378, 1440, 2247, 2266, 2294, 2348, 2434, 2443, 2458, 2464, 2771, 2772, 2784, 3074, 3195, 3196, 3208

 \int_zero:N 3140, 3504

 \c_one_int . 1557, 1997, 2756, 2775, 2781, 2787, 2791, 2794, 3180, 3199, 3205, 3211, 3215, 3218

 \c_zero_int 1418, 1765, 1775, 1787, 1788, 1803, 1805, 2339, 2895, 2971, 2981, 3151, 3515

\item 27, 37, 38, 56, 65, 78, 80, 81, 83, 91

\item 65, 66, 85, 87, 92, 94, 225, 1575, 1581, 1606, 1614, 1701, 1935, 1938, 2106, 2140, 2873, 2875, 3296, 3298, 3407

\item* 5, 11, 2138

item-pos* 2012

item-sym* 2012

\itemindent 22, 69

\itemindent 68

itemindent 613

\itemsep 79, 80

\itemsep 2709, 2715

\itemwidth 2763, 2807, 2811, 3187, 3231, 3235

K

keyans 11, 2517

keyans* 11, 3285

keyanspic 12, 2670

Keys for environments provide by enumext:

 above* 23, 48, 49

 above 23, 48, 49, 73, 76, 84, 92

 after 37, 38, 75, 78, 85, 92

 align 23, 30, 67, 87

 before* 37, 38, 73, 84, 92

 before 37, 38, 76

 below* 23, 48, 49

 below 23, 48, 49, 75, 78, 85, 92

 check-ans 23, 24, 26, 50, 52, 57, 63, 65, 66, 73-75, 89, 100

 columns-sep* 24, 54, 72, 84

 columns-sep 39, 55, 72, 73, 77, 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, 83

 minirigth* 26

 minirigth 26

 no-store 24, 51-53

 noitemsep 34, 43

 nosep 34, 43

 parindent 70

 parsep 34, 70, 87

 partopsep 34

 ref 25, 31

 resume 22, 50, 51, 69, 75, 85

 rightmargin 35

 save-ans 23, 50, 51, 55, 57, 61, 62, 66, 75, 78, 85, 92, 96

 save-key 24

 show-ans 24, 53, 54, 56, 58, 60, 67

 show-length 26, 36, 69, 99

 show-pos 24, 53, 54, 56, 58, 60, 67

 start 23, 26, 33, 34, 69

 store-brk 57, 58

 store-ref 24, 28, 53, 57, 59, 61, 63, 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	336, 358, 387, 468, 488, 504, 551, 571, 615, 634, 691, 700, 779, 796, 1215, 1324, 1333, 1385, 1482, 1511, 1529, 1647, 2014, 3548, 3611
\l_keys_key_str	3702
\keys_set:nn	350, 803, 1220, 1225, 1693, 2347, 2556, 2905, 3324, 3613, 3614, 3615, 3616, 3617, 3618, 3619, 3620, 3621, 3622, 3623, 3624, 3662
L	
label	466, 486, 504
Labels provide by enumext:	
\Alph*	29
\Roman*	29
\alph*	29
\arabic*	29, 31
\roman*	29
\labelsep	80
\labelsep	2710, 2713
labelsep	334
\labelwidth	29, 80
\labelwidth	2710, 2711
labelwidth	334
\leftmargin	22, 69
\leftmargin	68, 2710
legacy commands:	
\legacy_if:nTF	3057, 3060, 3444, 3447
\legacy_if_gset_false:n	893
\legacy_if_set_false:n	3059, 3446
\legacy_if_set_true:n	3019, 3044, 3051, 3064, 3371, 3395, 3451
\linewidth	73, 76
\linewidth	2418, 2566, 2732, 2759, 2820, 3183, 3244
\list	27
\list	223
list-indent	613
list-offset	613
\listparindent	2712
listparindent	613
\lrbox	3084, 3457
M	
\makebox	81
\makebox	1637, 1639, 2160, 3098, 3106, 3110, 3470, 3474
\makelabel	65, 67, 68, 81
\makelabel	67, 68, 2166, 2182
\makesavenoteenv	256
mark-ans	1480
mark-pos	1480, 1509
mark-ref	1480
mini-env	777
mini-sep	777
\minipage	27
\minipage	229
\miniright	9, 46, 1161, 2487, 2619
\miniright*	9
mode commands:	
\mode_if_vertical:TF	848, 876, 999, 1078
\mode_leave_vertical:	646, 660, 672, 684, 1606, 1614, 1635, 2158, 3096

msg commands:	
\msg_error:nn	2547, 2551, 2638, 2697, 2889, 3311, 3317, 3625
\msg_error:nnn	1165, 1169, 1194, 1211, 3538, 3543, 3608, 3678
\msg_error:nnnn	1661, 1665, 1669, 2539, 2634, 2642
\msg_fatal:nn	2334
\msg_fatal:nnn	288
\msg_info:nnn	13, 16, 238, 252
\msg_line_context:	3706, 3711, 3716, 3731, 3754, 3758, 3762, 3767, 3772, 3777, 3782, 3786, 3791, 3796, 3800, 3805, 3809, 3814, 3819, 3824, 3828, 3832
\msg_new:nnn	3679, 3683, 3687, 3691, 3696, 3700, 3704, 3709, 3714, 3729, 3744, 3751, 3756, 3760, 3765, 3770, 3775, 3780, 3784, 3789, 3794, 3798, 3803, 3807, 3812, 3817, 3822, 3826, 3830
\msg_term:nnn	1455, 1473
\msg_term:nnnn	2256, 2266, 2301, 2306
\msg_warning:nn	2486, 2618
\msg_warning:nnn	1458, 1476
\msg_warning:nnnn	1972, 2198, 2203, 2770, 2783, 3194, 3207
\multicolsep	73, 77
\multicolsep	2448, 2591

N	
\NeedsTeXFormat	3
\newcounter	291
\NewDocumentCommand	1161, 1657, 2630, 3522, 3576, 3632
\NewDocumentEnvironment	2311, 2517, 2670, 2861, 3285
\newlabel	28
\newlabel	274
no-store	1383
\noindent	83, 91
\noindent	2425, 2571, 2829, 2874, 3139, 3253, 3297, 3503
\nointerlineskip	2425, 2571, 2829, 3253
noitemsep	569
\nopagebreak	859, 887, 1010, 1089, 1152, 1158
\normalfont	1850, 1994, 3424
nosep	569

P	
Packages:	
enumext	21, 50, 68, 69, 78, 99
enumitem	28, 29
expl3	81
footnotehyper	27
hyperref	24, 26-28, 31, 59, 63, 87, 94, 99
lua-visual-debug	41
multicol	21, 99
shortlst	81
\par	859, 887, 1010, 1089, 1152, 1158, 1187, 1204, 1829, 2469, 2491, 2608, 2623, 2744, 2847, 2854, 3139, 3153, 3271, 3278, 3503, 3517
\parindent	3116, 3480
\parsep	40, 43, 79, 80
\parsep	1607, 1615, 2286, 2709, 2716, 2721
parsep	569
\parskip	3117, 3481
\partopsep	80
\partopsep	2287, 2714
partopsep	569
peek commands:	
\peek_meaning:NTF	2996, 3010, 3027, 3038, 3348, 3362, 3379
\peek_meaning_remove:NTF	3003, 3355

\peek_remove_spaces:n 2144

\phantomsection 28

\phantomsection 263

prg commands:

\prg_do_nothing: 267

\prg_new_protected_conditional:Npnn ... 201

\prg_replicate:nn 210, 3749

\prg_return_false: 205

\prg_return_true: 204

\printkeyans 13, 96, 3576

prop commands:

\prop_count:N .. 1557, 1817, 1853, 1916, 1997, 3427

\prop_gput_if_not_in:Nnn 1552, 1555

\prop_if_exist:NTF 1343, 3542

\prop_item:Nn 3545

\prop_new:N 1345

\ProvidesExplPackage 4

R

\raggedcolumns 2457, 2597

\ref 59, 61

ref 466, 486

\refstepcounter 3066, 3453

regex commands:

\regex_match:nnTF 203, 526, 528, 540, 542, 2358, 2371, 2916, 2929

\regex_replace_once:nnN 402

\renewcommand 414, 435, 445

\RenewDocumentCommand ... 2036, 2106, 2140, 2166, 2182

\RequirePackage 17

resume 1324

resume* 1324

rightmargin 613

\Roman 29, 33, 34

\Roman 310

\roman 29, 33, 34

\roman 311, 484, 3564

S

save-ans 1324

scan commands:

\scan_stop: 80, 2723, 2873, 3296, 3533, 3536

seq commands:

\seq_clear:N 3639

\seq_const_from_clist:Nn 3627

\seq_count:N 2683, 3643

\seq_gclear:N 2034, 2035

\seq_gput_right:Nn 1564, 2047, 2048

\seq_if_empty:NTF 2053, 3591, 3657

\seq_if_exist:NTF 1347, 3589

\seq_item:Nn 2741

\seq_map_function:NN 3648

\seq_map_inline:Nn .. 3597, 3602, 3636, 3658, 3659

\seq_map_pairwise_function:NNN 2055

\seq_new:N 94, 95, 112, 142, 143, 1349

\seq_pop_left:NN 3647

\seq_put_right:Nn 2644, 3655, 3672

\seq_set_from_clist:Nn 3640

\seq_set_map_e:NNn 3649

\seq_show:N 3593

\setcounter .. 537, 541, 543, 2244, 2246, 2291, 2293, 2688

\setenumext .. 5-8, 97, 3552, 3557, 3562, 3567, 3572, 3632

\setlength 1608, 1616

show-ans 1480, 1509

show-length 689

skip commands:

\skip_add:Nn . 828, 834, 840, 850, 854, 878, 882, 979, 985, 991, 1001, 1005, 1027, 1080, 1084, 2709

\skip_eval:n 1607, 1615

\skip_gset:Nn 1100, 1104, 1108

\skip_gzero_new:N 1095, 1096

\skip_horizontal:N 661, 673, 685, 3099, 3113, 3477

\skip_horizontal:n 647, 1636, 1644, 2159, 2161, 3097

\skip_if_eq:nnTF . 826, 832, 838, 902, 936, 977, 983, 989, 1020, 1025, 1046, 1097, 1119, 1232, 1246, 1260, 1271, 1282, 1293, 1304, 1315

\skip_new:N 54, 55, 59, 60, 61, 62, 63, 116, 172

\skip_set:Nn . 811, 815, 864, 868, 905, 909, 913, 920, 924, 928, 939, 944, 948, 954, 959, 964, 1022, 1023, 1024, 1031, 1035, 1039, 1048, 1053, 1057, 1060, 1064, 1068, 1099, 1103, 1121, 1125, 1129, 1135, 1139, 1143, 2703, 2717

\skip_set_eq:NN 2239, 2285, 2286, 3116, 3117, 3480, 3481

\skip_use:N 813, 817, 852, 856, 860, 880, 884, 903, 922, 931, 937, 942, 946, 957, 961, 962, 967, 1003, 1007, 1033, 1233, 1237, 1240, 1247, 1251, 1254, 2469

\skip_zero:N 2287, 2448, 2591, 2714, 2715

\skip_zero_new:N 1015, 1016, 1017, 1094, 1116, 1117, 1118

\c_zero_skip . 826, 832, 838, 903, 937, 977, 983, 989, 1020, 1025, 1046, 1097, 1119, 1233, 1247, 1260, 1271, 1282, 1293, 1304, 1315

\small 3554, 3559, 3564, 3569, 3574

\star 2018

start 549

\stepcounter 2040, 2651

store-ref 1480

str commands:

\c_backslash_str 3758, 3767, 3768, 3772, 3773, 3777, 3778, 3809, 3810, 3814, 3819, 3820

\c_colon_str 1816, 1915, 3533

\str_count:n 210, 3749

\str_if_eq:nnTF 2249, 2297

\str_if_eq_p:nn 2242, 2290

\str_if_in:nnTF 3529

\str_new:N 111, 167

\str_set:Nn ... 390, 391, 392, 1491, 1492, 1514, 1515

\string 256

\strutbox 907, 911, 915, 926, 930, 941, 950, 956, 966, 979, 985, 991, 1022, 1023, 1024, 1027, 1037, 1041, 1050, 1057, 1062, 1070, 1099, 1100, 1103, 1110, 1123, 1131, 1137, 1145, 2719

T

TeX and L^AT_EX 2_ε commands:

\@auxout 272

\protected@write 272

text commands:

\text_expand:n 3525

\textasteriskcentered 1488, 1502

\thepage 278

tl commands:

\c_space_tl 1873, 1942, 1987, 2009, 3388, 3389, 3716, 3731

\tl_clear:N 363, 369, 1690, 1863, 1932, 3385

\tl_clear_new:N 320

\tl_const:Nn 144, 304

\tl_gclear:N 1974, 2177, 2515, 2858, 3100, 3168, 3282

\tl_gput_right:Nn 305

<code>\tl_greplace_all:Nnn</code>	326
<code>\tl_gset:Nn</code>	1962, 2477, 2962, 3033
<code>\tl_gset_eq:NN</code>	322, 2086, 3093
<code>\tl_if_blank:nTF</code>	3091, 3401, 3411, 3432
<code>\tl_if_empty:nTF</code> .	420, 454, 460, 1571, 1602, 1716, 1970, 2156, 3670
<code>\tl_if_novalue:nTF</code> ..	1691, 1702, 1871, 1940, 1986, 2008, 2038, 2063, 2082, 2087, 2117, 2681, 2903, 3322, 3386, 3634
<code>\tl_map_inline:Nn</code>	323, 400
<code>\tl_new:N</code> 32, 39, 41, 42, 75, 76, 77, 83, 84, 85, 87, 88, 89, 91, 92, 98, 99, 109, 110, 121, 133, 134, 135, 138, 146, 147, 150, 151, 166, 169	
<code>\tl_put_left:Nn</code> 1579, 1612, 1699, 1980, 2002, 3400, 3407	
<code>\tl_put_right:Nn</code> 321, 412, 433, 443, 1533, 1540, 1583, 1618, 1701, 1707, 1715, 1718, 1728, 1733, 1736, 1742, 1768, 1778, 1792, 1808, 1814, 1819, 1866, 1869, 1873, 1900, 1905, 1910, 1913, 1922, 1935, 1938, 1942, 1952, 1987, 2009, 2363, 2376, 2921, 2934, 3403, 3413, 3434, 3550, 3555, 3560, 3565, 3570	
<code>\tl_remove_all:Nn</code>	3669
<code>\tl_remove_once:Nn</code>	1756, 1885
<code>\tl_replace_all:Nnn</code>	325
<code>\tl_reverse:N</code>	1755, 1757, 1884, 1886
<code>\tl_set:Nn</code> 152, 290, 364, 368, 373, 374, 408, 427, 644, 658, 670, 682, 1342, 1487, 1501, 1847, 1991, 2084, 3388, 3389, 3421, 3667	
<code>\tl_set_eq:NN</code> 331, 409, 411, 430, 432, 440, 442, 1754, 1883, 1896, 2129, 2133, 2662, 2664	
<code>\tl_to_str:n</code>	3525
<code>\tl_trim_spaces:n</code>	321, 3655, 3667, 3673
<code>\tl_use:N</code> .	327, 330, 422, 456, 462, 715, 719, 723, 727, 731, 735, 739, 743, 747, 751, 755, 759, 763, 767, 771, 775, 1641, 1761, 1769, 1780, 1794, 1799, 1811, 2071, 2077, 2102, 2120, 2124, 2132, 2160, 2168, 2169, 2176, 2184, 2185, 2191, 2318, 2523, 2667, 2852, 3103, 3114, 3118, 3276, 3414, 3435, 3467, 3478, 3482, 3579, 3580, 3581, 3582, 3583, 3651
token commands:	
<code>\token_to_str:N</code>	274
<code>\topsep</code>	1608, 1616
<code>topsep</code>	<u>569</u>
<code>\typeout</code>	242, 245, 255, 256
U	
<code>\u</code>	403
use commands:	
<code>\use:N</code>	211, 2173, 2320
<code>\use:n</code>	3531
<code>\use_none:nn</code>	266
<code>\usecounter</code>	2240, 2288
V	
<code>\value</code>	2497, 2502, 2952, 2957
<code>\vspace</code> 894, 1237, 1240, 1251, 1254, 1264, 1266, 1275, 1277, 1286, 1288, 1297, 1299, 1308, 1310, 1319, 1321, 1607, 1615, 2678, 2689, 3154, 3518	
W	
<code>widest</code>	<u>549</u>
<code>wrap-ans</code>	<u>1480</u>
<code>wrap-label</code>	<u>334</u>
<code>wrap-label*</code>	<u>334</u>