

enumext

ENUMERATE EXERCISE SHEETS

V1.0 2024-04-03^{*}

©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	8
1.1	Description	3	4.1	Keys for storage	9
1.2	Installation and usage	3	4.2	Keys for internal label and ref	9
1.3	User interface	3	4.3	Keys for check answers	9
1.4	Internal counters	3	4.4	The command <code>\anskey</code>	9
1.5	The label and ref system	3	4.5	The environment <code>keyans</code>	10
1.6	The concept of left margin	3	4.5.1	The <code>\item*</code> in <code>keyans</code>	10
1.7	Support for <code>multicols</code>	4	4.6	The environment <code>keyanspic</code>	11
1.8	Support for <code>minipage</code>	4	4.6.1	The command <code>\anspic</code>	11
2	The environment <code>enumext</code>	4	4.7	Printing stored content	12
2.1	The <code>\item*</code> in <code>enumext</code>	5	4.7.1	The command <code>\getkeyans</code>	12
2.1.1	Keys for <code>\item*</code> in <code>enumext</code>	5	4.7.2	The command <code>\printkeyans</code>	12
3	The command <code>\setenumext</code>	5	5	Full examples	13
3.1	Keys for <code>label</code> and <code>ref</code>	5	6	The way of non-enumerated lists	15
3.2	Keys for spaces	6	6.1	Fake <code>itemize</code> environment	15
3.2.1	Vertical spaces	6	6.2	Fake <code>description</code> environment	15
3.2.2	Horizontal spaces	7	7	Final notes	16
3.3	Keys for <code>add code</code>	7	8	References	17
3.4	Keys for <code>start</code> and <code>resume</code>	8	9	Change history	17
3.5	Keys for <code>multicols</code>	8	10	Index of Documentation	18
3.6	Keys for <code>minipage</code>	8	11	Implementation	19
3.6.1	The command <code>\miniright</code>	8	12	Index of Implementation	92

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 multicols, 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 2024-04-03.

[†]E-mail: pablgonz@educarchile.cl.

1 Introduction

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

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

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

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

3. True False

(a) $\alpha > \delta$

(b) \LaTeX 2e is cool?

4. Related to Linux
- (a) You use linux?

(b) Usually uses the package manager?

(c) Rate the following package and class

i. `xsim-exam`

ii. `xsim`

iii. `exsheets`

Sometimes we are also interested in showing the “answers” along with the questions:

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

* `(x - 1)^2`

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

* `3(x + y + z)`

3. True False

(a) $\alpha > \delta$

* `False`

(b) \LaTeX 2e is cool?

* `Very True!`

4. Related to Linux
- (a) You use linux?

* `Yes`

(b) Usually uses the package manager?

* `Yes, dnf`

(c) Rate the following package and class

i. `xsim-exam`

* `doesn't exist for now :(`

ii. `xsim`

* `very good`

iii. `exsheets`

* `obsolete`

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

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

Or we are interested in printing all the “answers”:

1. $(x - 1)^2$

2. $3(x + y + z)$

3. (a) False

(b) Very True!

4. (a) Yes
- (b) Yes, dnf

(c) i. doesn't exist for now :(

ii. very good

iii. obsolete

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

1. First type of questions

(A) value

(B) correct

2. Second type of questions

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

II. $\alpha = \delta$

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

(A) I only

(B) II only

(C) I and II only

(D) I and III only

(E) I, II, and III

★ 3. Third type of questions

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

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

(A) value

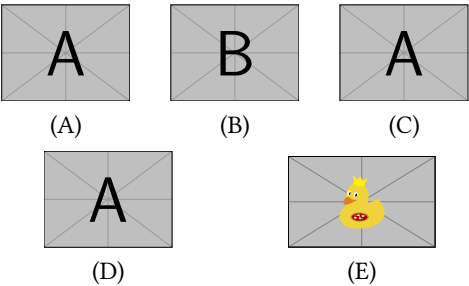
(B) value

(C) value

(D) value

(E) value

4. Question with image and label below:



5. Question with image on left side:



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

1. (B) $x = 5$

2. (D)

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

5. (D) “other note”

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

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

1.1 Description

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.

1.2 Installation and usage

This package can be used with `xelatex`, `lualatex`, `pdflatex` and the classical `latex>dvips>ps2pdf` and is present in \TeX Live and \MiKTeX , use the package manager to install. For manual installation, download `enumext.zip` and unzip it, run `lualatex enumext.dtx` and move all files to appropriate locations, then run “`mktexlsr`”. To produce the documentation run `lualatex enumext.dtx` two times.

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

The package is loaded in the usual way:

```
\usepackage{enumext}
```

1.3 User interface

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

1.4 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 and the `enumXvi` counter for the `keyanspic` 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.

1.5 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 \LaTeX `\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` key (§4.2) when the `save-ans` key (§4.1) is active.

1.6 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.

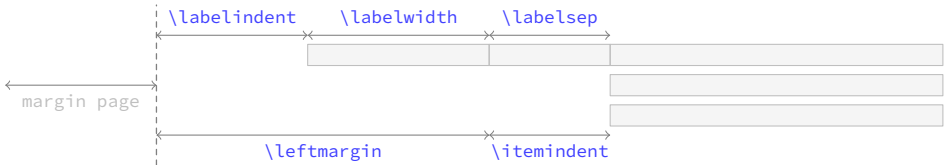


Figure 1: Representation of horizontal lengths in `enumitem`.

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.

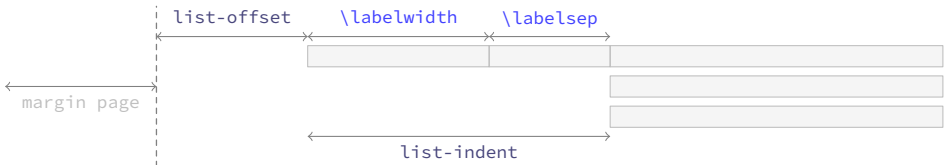


Figure 2: Representation of horizontal lengths concept in `enumext`.

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.

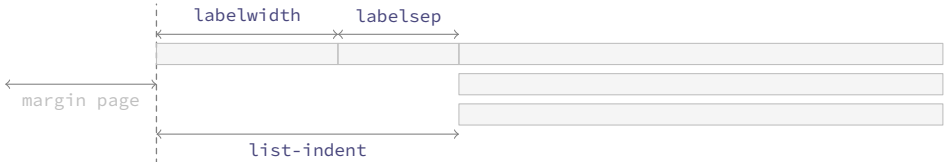


Figure 3: Default horizontal lengths `list-offset=0pt`, `list-indent=\labelwidth+\labelsep` in `enumext`.

1.7 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.

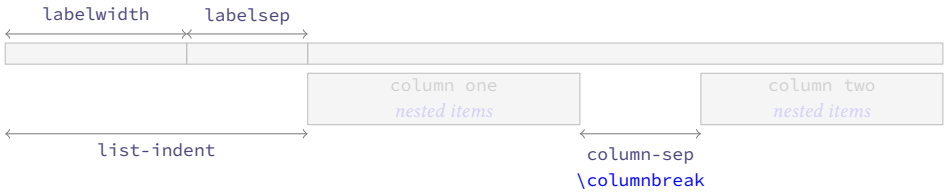


Figure 4: Representation of the two column output for a nested level in `enumext` environment.

The “*non starred*” version of the `multicol` 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.

1.8 Support for minipage

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



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

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

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 `LaTeX`, `\item` and `\item [<custom>]` commands work in the usual way.

The environment can be nested with at most “*four levels*” and the options can be configured globally using `\setenumext` command and locally using [`<key = val>`] in the environment.

Example

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

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

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

• It should be kept in mind that using any *key* that sets a *rubber or rigid lengths* for vertical or horizontal space on a level will influence the vertical and horizontal space for *inners levels* and `keyans` and `keyanspic` environments. All *keys* related to vertical or horizontal spacing accept a “skip” or “dim” expression if passed between braces, i.e. you do not need to use `\dimexpr` or `\dimeval` to perform calculations.

3.1 Keys for label and ref

`label` = {`\alph*` | `\Alph*` | `\arabic*` | `\roman*` | `\Roman*`} default: *by levels*
 Sets the *label* that will be printed at the *current level*. The default value for first level are `\arabic*`, for second level are `(\alph*)`, for third level are `\roman*`, and for fourth level are `\Alph*`.
 • This key is intended to give the basic structure with which the *label* will be displayed, and the and the form in which it is used by standard “label and ref” and the “internal reference” system with the `store-ref` key. You cannot use commands with *label* as an argument, for example `\emph{\alph*}` will return an error. For full customization of how *label* is displayed use the `font` or `wrap-label` keys.
`ref` = {*code* {`\alph*` | `\Alph*` | `\arabic*` | `\roman*` | `\Roman*`} *more code*} default: *empty*
 Modifies the way *cross references* are displayed. The `label` key sets the default form of the *cross references*, by using this key you can define a different format, for example: `ref=\emph{\alph*}` is valid.

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

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

`labelsep = {<rigid length>}` default: 0.3333em

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

`labelwidth = {<rigid length>}` default: by label

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

`widest = {<integer | string>}` default: empty

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

`font = {}` default: empty

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

`align = {<left | right | center>}` default: left

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

`wrap-label = {<code {#1} more code>}` default: empty

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

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

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

`wrap-label* = {<code {#1} more code>}` default: empty

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

3.2 Keys for spaces

`show-length = {<true | false>}` default: false

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

3.2.1 Vertical spaces

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

3.2.2 Horizontal spaces

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

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

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

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

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

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

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

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

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

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

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

3.3 Keys for add code

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

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

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

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

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

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

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

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

3.4 Keys for start and resume

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

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

3.5 Keys for multicol

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

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

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

3.6 Keys for minipage

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

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

3.6.1 The command `\miniright`

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

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

4 The storage system

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

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

4.1 Keys for storage

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

4.2 Keys for internal label and ref

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

4.3 Keys for check answers

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

4.4 The command `\anskey`

`\anskey` `\anskey{⟨content⟩}`

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

Example

- | | |
|--|--|
| <p>★ 1. Text containing our instructions or questions.</p> <p style="margin-left: 20px;">* first answer</p> <p>2. Text containing our instructions or questions.</p> <p style="margin-left: 20px;">(a) Question.</p> <p style="margin-left: 40px;">* second answer</p> | <p>3. Text containing our instructions or questions.</p> <p style="margin-left: 20px;">* third answer</p> <p>4. Text containing our instructions or questions.</p> <p style="margin-left: 20px;">* fourth answer</p> |
|--|--|

```
\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.

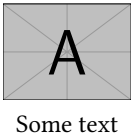
(A) Choice

(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.

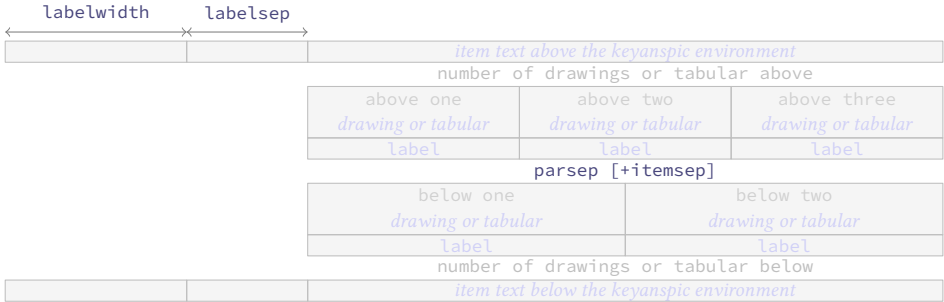


Figure 6: Representation of the `keyanspic` environment with optional argument `[3,2]` in `enumext`.

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}`

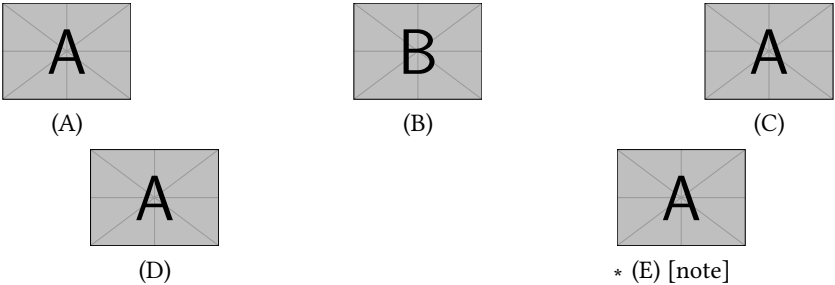
The `\anspic` command take three arguments, the *starred version* “*” store the current `\label` next to the `\content` (if it is present) in `\store name` set by `save-ans` key.

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. $3(x + y + z)$

2. (a) Very True!

3. (a) Yes

(b) i. very good

ii. obsolete
- *

*

*

*

*


5 Full examples

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

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

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

Example 1

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

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

A

 36 km/h.

B

 360 km/h.

C

 27,8 km/h.

D

 $3,60 \times 10^8$ km/h.
2. In fisica nucleare si usa l'angstrom (simbolo: $1 \text{ \AA} = 1 \times 10^{-10}$ m) e il fermi o femtometro ($1 \text{ fm} = 1 \times 10^{-15}$ m). Qual è la relazione tra queste due unità di misura?

A

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

B

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

C

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

D

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

A

 36 km/h.

B

 360 km/h.

C

 27,8 km/h.

D

 $3,60 \times 10^8$ km/h.
4. In fisica nucleare si usa l'angstrom (simbolo: $1 \text{ \AA} = 1 \times 10^{-10}$ m) e il fermi o femtometro ($1 \text{ fm} = 1 \times 10^{-15}$ m). Qual è la relazione tra queste due unità di misura?

A

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

B

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

C

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

D


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

2. A

3. B

4. A

Example 2

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

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

A

 36 km/h.

☒ B

 360 km/h.

C

 27,8 km/h.

D

 $3,60 \times 10^8$ km/h.
2. In fisica nucleare si usa l'angstrom (simbolo: $1 \text{ \AA} = 1 \times 10^{-10}$ m) e il fermi o femtometro ($1 \text{ fm} = 1 \times 10^{-15}$ m). Qual è la relazione tra queste due unità di misura?

☒ A

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

B

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

C

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

D

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

A

 36 km/h.

☒ B

 360 km/h.

C

 27,8 km/h.

D

 $3,60 \times 10^8$ km/h.
4. In fisica nucleare si usa l'angstrom (simbolo: $1 \text{ \AA} = 1 \times 10^{-10}$ m) e il fermi o femtometro ($1 \text{ fm} = 1 \times 10^{-15}$ m). Qual è la relazione tra queste due unità di misura?

☒ A

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

B

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

C

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

D

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

2. A

3. B

4. A

*

*

*

*

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

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

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

A value

B value

C value
4. Question with image and label below:



A



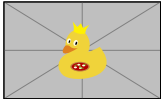
D



B



C



E

5. Question with image on left side:
- A value

B value

C value

D correct

E value



Test keys

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

- * 4. B
- * 5. D other note
- *

*
*
*

Example 4

A “simple worksheet” using ducks :) 📄.

- 1

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

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

True False
- (a)

 $\alpha > \delta$
- (b)

~~ETX~~ze is cool?

4

Related to Linux

(a)

You use linux?

(b)

Usually uses the package manager?

(c)

Rate the following package and class

i.

`xsim-exam`

ii.

`xsim`

iii.

`exsheets`

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

1. $(x - 1)^2$
2. $3(x + y + z)$
3. (a) False
- (b)

Very True!
4. (a) Yes
- (b)

Yes, dnf
- (c)

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

*
*
*
*
*

Example 5

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

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

1. A)

2. C)

3. B)

4. D)

6 The way of non-enumerated lists

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

6.1 Fake itemize environment

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

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

6.2 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,

LoNg vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris.

7 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`[9] and `l3seq`[9] 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.

8 References

- [1] HIRSCHHORN, PHILIP. “Using the exam document class”. Available from CTAN, <https://www.ctan.org/pkg/exam>, 2023.
- [2] NIEDERBERGER, CLEMENS. “xsim – eXercise Sheets IMproved”. Available from CTAN, <https://www.ctan.org/pkg/xsim>, 2023.
- [3] MITTELBACH, FRANK. “An environment for multicolumn output”. Available from CTAN, <https://www.ctan.org/pkg/multicol>, 2023.
- [4] The \LaTeX Project. “enumerate – Enumerate with redefinable labels”. Available from CTAN, <https://www.ctan.org/pkg/enumerate>, 2023.
- [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>, 2023.
- [7] The \LaTeX Project. “Extensive support for hypertext in \LaTeX ”. Available from CTAN, <https://www.ctan.org/pkg/hyperref>, 2023.
- [8] The \LaTeX Project. “The expl3 package”. Available from CTAN, <https://www.ctan.org/pkg/l3kernel>, 2023.
- [9] The \LaTeX Project. “The \LaTeX 3 Interfaces”. Available from CTAN, <https://www.ctan.org/pkg/l3kernel>, 2023.
- [10] The \LaTeX Project. “The xparse package”. Available from CTAN, <https://www.ctan.org/pkg/xparse>, 2023.
- [11] GUNDLACH, PATRICK. “The lua-visual-debug package”. Available from CTAN, <https://www.ctan.org/pkg/lua-visual-debug>, 2023.
- [12] LEMVIG, MOGENS. “The shortlst package”. Available from CTAN, <https://www.ctan.org/pkg/shortlst>, 1998.
- [13] NIEDERBERGER, CLEMENS. “tasks – Horizontally columned lists”. Available from CTAN, <https://www.ctan.org/pkg/tasks>, 2022.

9 Change history

v1.0 2024-04-03 – First public release.

10 Index of Documentation

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

C

Document class:

article1

book1

exam2

letter1

report1

\columnbreak4

\columnsep8

Commands provide by enumext:

\anskey3, 8–10

\anspic*3, 9, 11

\anspic11

\getkeyans3, 9, 12

\item*3–6, 9, 10

\item4, 6, 8–10

\miniright3, 4, 8

\printkeyans3, 5, 9, 12

\setenumext3–6, 9, 10, 12

Counters defined by enumext:

enumXiii3

enumXii3

enumXiv3

enumXi3

enumXvi3

enumXv3

E

Environments provide by enumext:

enumext3–5, 8–10, 12, 15

keyanspic3, 5, 8, 9, 11, 15

keyans3–6, 8–11, 15

Environments:

enumerate1–4, 17

list3, 7, 17

minipage2–4, 8, 17

multicols2, 4, 8

I

\item3, 4

\itemsep7

K

Keys for environments provide by enumext:

above*7

above7

after8

align6, 16

before*7

before7

below*7

below7

check-ans9

columns-sep4, 8

columns4, 7, 8

first8

font5, 6

item-pos*5

item-sym*5

itemindent7

itemsep7, 11

labelsep4–9, 16

labelwidth4–9, 16

label5, 6, 8, 10, 12, 15, 16

list-indent3, 4, 7

list-offset3, 4, 7, 16

listparindent7

mark-ans9

mark-pos9

mark-ref9

mini-env4, 7, 8

mini-sep4, 8

no-store9

noitemsep7

nosep7, 15

parsep6, 7, 11

partopsep6

ref3, 5, 6

resume8

rightmargin7

save-ans3, 8–12

show-ans9

show-length6

show-pos9, 12

start8

store-ref3, 5, 9, 12

topsep6, 7

widest6

wrap-ans9

wrap-label*6, 16

wrap-label5, 6

L

\label3

Labels provide by enumext:

\Alph*5, 6, 10

\Roman*5, 6

\alph*5, 6

\arabic*5, 6

\roman*5, 6

\labelsep3, 4, 6

\labelwidth3, 4, 6

\linewidth8

\listparindent7

P

Packages:

enumerate16

enumext1–4, 11, 16, 17

enumitem3, 7, 16, 17

hyperref9, 17

l3prop1, 17

l3seq1, 17

multicol1, 4, 17

xsim2

\parsep6

\partopsep6

R

\raggedcolumns4

\ref3

\rightmargin7

T

\topsep6

11 Implementation

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

11.1 General conventions

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

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

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

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

11.2 Initial set up

Start the DocStrip guards.

```
1 <{*package>
```

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

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

11.3 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]
```

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

```
4 \IfPackageLoadedTF { multicol } { }
5 {
6   \RequirePackage{ multicol }[2023-03-30]
7 }
```

Finally we declare the `enumext` package.

```
8 \ProvidesExplPackage
9   {enumext}
10  {2024-04-03}
11  {1.0}
12  {Enumerate exercise sheets}
```

11.4 Definition of variables

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

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.

```
13 \int_new:N \__enumext_level_int
14 \int_new:N \__enumext_level_h_int
15 \int_new:N \__enumext_keyans_level_int
16 \int_new:N \__enumext_keyans_level_h_int
17 \int_new:N \__enumext_keyans_pic_level_int
18 \int_new:N \__enumext_starred_bool
19 \bool_new:N \g__enumext_starred_bool
20 \bool_new:N \l__enumext_standar_bool
21 \bool_new:N \g__enumext_standar_bool
22 \bool_new:N \l__enumext_keyans_env_bool
```

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

```

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

```

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

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

The initial values of these variables are set by the function `__enumext_define_counters:Nn` and then modified by the function `__enumext_label_style:Nnn` used by `label` key (§11.8).

```

23 \cs_set_protected:Npn \__enumext_tmp:n #1
24 {
25   \tl_new:c { l__enumext_counter_#1_tl }
26 }
27 \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` (§11.21). The global token list `\g__enumext_item_symbol_tl` is used by `item-sym*` key (§11.26).

```

28 \bool_new:N \l__enumext_resume_bool
29 \int_new:N \g__enumext_resume_int
30 \bool_new:N \l__enumext_resume_vii_bool
31 \int_new:N \g__enumext_resume_vii_int
32 \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` (§11.12) and `label` (§11.10) keys.

```

33 \dim_new:N \l__enumext_current_widest_dim
34 \tl_new:N \g__enumext_counter_styles_tl
35 \tl_new:N \g__enumext_widest_label_tl
36 \box_new:N \l__enumext_label_width_by_box

```

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

```

\l__enumext_leftmargin_tmp_X_bool
\l__enumext_leftmargin_tmp_X_dim
\l__enumext_leftmargin_X_dim
\l__enumext_itemindent_X_dim

```

The boolean variable `\l__enumext_leftmargin_tmp_X_bool` and the dimensional variable `\l__enumext_leftmargin_tmp_X_dim` are used by the `list-indent` key (§11.14).

The variables `\l__enumext_leftmargin_X_dim` and `\l__enumext_itemindent_X_dim` are used (and set) by the function `__enumext_calc_hspace:NNNNNNNNNN` (§11.30) which determines the internal values for `\leftmargin` and `\itemindent`.

```

37 \cs_set_protected:Npn \__enumext_tmp:n #1
38 {
39   \bool_new:c { l__enumext_leftmargin_tmp_#1_bool }
40   \dim_new:c { l__enumext_leftmargin_tmp_#1_dim }
41   \dim_new:c { l__enumext_leftmargin_#1_dim }
42   \dim_new:c { l__enumext_itemindent_#1_dim }
43 }
44 \clist_map_inline:nn { i, ii, iii, iv, v, vi, vii, viii } { \__enumext_tmp:n {#1} }

```

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

```

\l__enumext_multicols_above_X_skip
\l__enumext_multicols_below_X_skip

```

Internal variables used by `columns` key §11.18).

```

45 \cs_set_protected:Npn \__enumext_tmp:n #1
46 {
47   \skip_new:c { l__enumext_multicols_above_#1_skip }
48   \skip_new:c { l__enumext_multicols_below_#1_skip }
49 }
50 \clist_map_inline:nn { i, ii, iii, iv, v } { \__enumext_tmp:n {#1} }

```

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

```

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

```

Internal variables used by `\miniright` command (§11.19.4) and the keys `miniright`, `miniright*`, `mini-env` and `mini-sep` (§11.17, §11.19).

```

51 \int_new:N \g__enumext_minipage_stat_int
52 \skip_new:N \l__enumext_minipage_left_skip
53 \skip_new:N \l__enumext_minipage_right_skip
54 \skip_new:N \l__enumext_minipage_after_skip
55 \skip_new:N \g__enumext_minipage_right_skip
56 \skip_new:N \g__enumext_minipage_after_skip
57 \cs_set_protected:Npn \__enumext_tmp:n #1

```

```

58   {
59     \dim_new:c { \__enumext_minipage_left_#1_dim }
60     \bool_new:c { \__enumext_minipage_active_#1_bool }
61   }
62   \clist_map_inline:nn { i, ii, iii, iv, v, vii, viii } { \__enumext_tmp:n {#1} }

```

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

```

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

```

The integer variable `__enumext_start_X_int` are used by the `start` key (§11.12), the token list `__enumext_fake_item_indent_X_tl` is used by `itemindent` key, the variables `__enumext_label_fill_left_X_tl` and `__enumext_label_fill_right_X_tl` are used by the `align` key (§11.10). The boolean vars `__enumext_vspace_a_star_X_bool`, `__enumext_vspace_b_star_X_bool` are used by `above`, `above*`, `below` and `below*` keys

```

63   \cs_set_protected:Npn \__enumext_tmp:n #1
64   {
65     \bool_new:c { \__enumext_wrap_label_#1_bool }
66     \bool_new:c { \__enumext_wrap_label_opt_#1_bool }
67     \int_new:c { \__enumext_start_#1_int }
68     \tl_new:c { \__enumext_fake_item_indent_#1_tl }
69     \tl_new:c { \__enumext_label_fill_left_#1_tl }
70     \tl_new:c { \__enumext_label_fill_right_#1_tl }
71     \bool_new:c { \__enumext_vspace_a_star_#1_bool }
72     \bool_new:c { \__enumext_vspace_b_star_#1_bool }
73   }
74   \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.)

```

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

```

The boolean variable `\l__enumext_store_active_bool` setting by `save-ans` key (§11.21) 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 (§11.21).

The variable `\l__enumext_store_anskey_arg_tl` stores the contents of `\anskey` (§11.24) and the variable `\l__enumext_store_keyans_label_tl` stores the contents of `\item*` (§11.28.2) for the `keyans` and `keyans*` environments and the contents of `\anspic*` (§11.34.1) for the `keyanspic` environment.

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

```

75   \bool_new:N \__enumext_store_active_bool
76   \tl_new:N \__enumext_store_name_tl
77   \tl_new:N \g__enumext_store_name_tl
78   \tl_new:N \__enumext_store_anskey_arg_tl
79   \int_new:N \__enumext_store_columns_join_int
80   \tl_new:N \__enumext_store_keyans_label_tl
81   \tl_new:N \__enumext_keyans_tmpa_tl

```

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

```

\__enumext_setkey_tmpa_tl
\__enumext_setkey_tmpp_tl
\__enumext_setkey_tmpa_int
\__enumext_setkey_tmpa_seq
\__enumext_setkey_tmpp_seq

```

Internal variables used by the command `\setenumext` (§11.38).

```

82   \tl_new:N \__enumext_setkey_tmpa_tl
83   \tl_new:N \__enumext_setkey_tmpp_tl
84   \int_new:N \__enumext_setkey_tmpa_int
85   \seq_new:N \__enumext_setkey_tmpa_seq
86   \seq_new:N \__enumext_setkey_tmpp_seq

```

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

```

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

```

Internal variables used by `[⟨key = val⟩]` in `enumext` and `enumext*` environment, the command `\printkeyans` (§11.37) and the keys `columns*` and `columns-sep*`.

```

87   \cs_set_protected:Npn \__enumext_tmp:n #1
88   {
89     \tl_new:c { \__enumext_store_opt_#1_tl }
90     \tl_new:c { \__enumext_print_keyans_#1_tl }
91     \bool_new:c { \__enumext_store_columns_#1_bool }
92     \int_new:c { \__enumext_store_columns_#1_int }
93     \bool_new:c { \__enumext_store_columns_sep_#1_bool }
94     \dim_new:c { \__enumext_store_columns_sep_#1_dim }
95     \bool_new:c { \__enumext_store_upper_level_#1_bool }
96   }
97   \clist_map_inline:nn { i, ii, iii, iv, vii } { \__enumext_tmp:n {#1} }

```

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

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

Internal variables for “*storage system*” mechanism used by `\anskey` (§11.24), `keyans` and `keyanspic` environments. These variables are used by `show-ans`, `show-pos`, `mark-ans`, `save-key` and `mark-ref` keys (§11.23).

```
98 \bool_new:N \l__enumext_show_answer_bool
99 \bool_new:N \l__enumext_show_position_bool
100 \tl_new:N \l__enumext_mark_ref_sym_tl
101 \tl_new:N \l__enumext_mark_answer_sym_tl
102 \str_new:N \l__enumext_mark_position_str
```

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

Internal variables used by `keyanspic` environment (§11.34.2).

```
103 \seq_new:N \l__enumext_keyans_pic_body_seq
104 \dim_new:N \l__enumext_keyans_pic_width_dim
105 \int_new:N \l__enumext_keyans_pic_above_int
106 \int_new:N \l__enumext_keyans_pic_below_int
107 \skip_new:N \l__enumext_keyans_pic_above_skip
```

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

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

Internal variables used by “*check answer*” mechanism (§11.22.1) controlled by the `check-ans` and `no-store` keys.

```
108 \bool_new:N \l__enumext_check_ans_bool
109 \bool_new:N \g__enumext_check_ans_show_bool
110 \bool_new:N \g__enumext_check_ans_show_h_bool
111 \tl_new:N \g__enumext_check_ans_item_tl
112 \int_new:N \l__enumext_compare_items_ans_int
113 \int_new:N \g__enumext_count_item_ans_int
114 \int_new:N \g__enumext_count_item_all_int
115 \cs_set_protected:Npn \__enumext_tmp:n #1
116 {
117   \int_new:c { g__enumext_count_level_#1_int }
118   \int_new:c { g__enumext_count_item_#1_int }
119 }
120 \clist_map_inline:nn { i, ii, iii, iv, vii } { \__enumext_tmp:n {#1} }
```

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

```
\l__enumext_hyperref_bool
\l__enumext_footnotes_key_bool
```

The boolean variable `\l__enumext_hyperref_bool` will determine if the `hyperref` package is present or load in memory (§11.7). The boolean variable `\l__enumext_footnotes_key_bool` determine if `hyperref` is load with key `hyperfootnotes=true`.

```
121 \bool_new:N \l__enumext_hyperref_bool
122 \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.

```
123 \tl_new:N \l__enumext_newlabel_arg_one_tl
124 \tl_new:N \l__enumext_newlabel_arg_two_tl
125 \tl_new:N \l__enumext_store_write_aux_file_tl
126 \cs_set_protected:Npn \__enumext_tmp:n #1
127 {
128   \tl_new:c { l__enumext_label_copy_#1_tl }
129 }
130 \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`.

```
131 \int_new:N \g__enumext_footnote_int
132 \seq_new:N \g__enumext_footnote_arg_seq
133 \seq_new:N \g__enumext_footnote_int_seq
```

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

Internal variables used by `ref` key (§11.17, §11.18).

```

134 \tl_const:Nn \c__enumext_counter_style_tl
135   { { arabic } { roman } { Roman } { alph } { Alph } }
136 \tl_new:N \l__enumext_ref_key_arg_tl
137 \tl_new:N \l__enumext_ref_aux_tl
138 \cs_set_protected:Npn \__enumext_tmp:n #1
139   {
140     \tl_new:c { l__enumext_counter_style_for_ref_#1_tl }
141     \tl_new:c { l__enumext_the_counter_#1_tl }
142     \tl_set:ce { l__enumext_the_counter_#1_tl } { \exp_not:c { theenumX#1 } }
143   }
144 \clist_map_inline:nn { i, ii, iii, iv, v, vi, vii, viii } { \__enumext_tmp:n {#1} }

```

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

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

```

145 \cs_set_protected:Npn \__enumext_tmp:n #1
146   {
147     \bool_new:c { l__enumext_item_starred_#1_bool }
148     \int_new:c { l__enumext_item_column_pos_#1_int }
149     \int_new:c { g__enumext_item_count_all_#1_int }
150     \int_new:c { l__enumext_joined_item_#1_int }
151     \int_new:c { l__enumext_joined_item_aux_#1_int }
152     \int_new:c { l__enumext_tmpa_#1_int }
153     \box_new:c { l__enumext_item_text_#1_box }
154     \dim_new:c { l__enumext_joined_width_#1_dim }
155     \dim_new:c { l__enumext_item_width_#1_dim }
156     \tl_new:c { g__enumext_item_symbol_aux_#1_tl }
157     \str_new:c { l__enumext_align_label_#1_str }
158     \bool_new:c { g__enumext_minipage_active_#1_bool }
159     \tl_new:c { g__enumext_miniright_code_#1_tl }
160     \bool_new:c { g__enumext_minipage_center_#1_bool }
161     \dim_new:c { g__enumext_minipage_right_#1_dim }
162     \skip_new:c { g__enumext_minipage_right_#1_skip }
163   }
164 \clist_map_inline:nn { vii, viii } { \__enumext_tmp:n {#1} }

```

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

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

```

165 \tl_const:Nn \c__enumext_all_envs_clist
166   {
167     {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv},
168     {keyans}{v}, {enumext*}{vii}, {keyans*}{viii}
169   }

```

(End of definition for `\c__enumext_all_envs_clist`.)

11.5 Some utility functions

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

```

170 \cs_new_protected:Npn \__enumext_at_begin_document:n #1
171   {
172     \hook_gput_code:nnn {begindocument} {enumext} { #1 }
173   }

```

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

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

```

174 \cs_new_protected:Npn \__enumext_after_env:nn #1 #2
175   {
176     \hook_gput_code:nnn {env/#1/after} {enumext} {#2}
177   }

```

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

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

```

178 \cs_new:Nn \__enumext_level:
179 {
180   \int_to_roman:n { \__enumext_level_int }
181 }

```

(End of definition for `__enumext_level:`.)

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

```

\__enumext_level_end:n
182 \cs_new:Npn \__enumext_level_set:n #1
183 {
184   \cs_set_eq:cN { \__enumext_level_#1: } \__enumext_level:
185   \cs_set:Nn \__enumext_level: { #1 }
186 }
187 \cs_new:Npn \__enumext_level_end:n #1
188 {
189   \cs_set_eq:Nc \__enumext_level: { __enumext_level_#1: }
190 }

```

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

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

```

191 \prg_new_protected_conditional:Npnn \__enumext_if_is_int:n #1 { T, F, TF }
192 {
193   \regex_match:nnTF { ^[\+|-]?[\d]+$ } {#1} % $
194   { \prg_return_true: }
195   { \prg_return_false: }
196 }

```

(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.

```

197 \cs_new:Npn \__enumext_show_length:nnn #1 #2 #3
198 {
199   * ~ #2
200   \prg_replicate:nn { 14 - \str_count:n {#2} } { ~ }
201   = ~ \use:c { #1_use:c } { \__enumext_#2_#3_#1 } \\
202 }

```

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

11.6 Copying list and minipage environments

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

```

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

```

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

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

```

203 \__enumext_at_begin_document:n
204 {
205   \cs_new_eq:NN \__enumext_start_list:nn \list
206   \cs_new_eq:NN \__enumext_stop_list: \endlist
207   \cs_new_eq:NN \__enumext_item_std:w \item
208 }

```

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

The `minipage` environment provided by \TeX 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.

```

209 \__enumext_at_begin_document:n
210 {
211     \cs_new_eq:NN \__enumext_minipage:w \minipage
212     \cs_new_eq:NN \__enumext_endminipage: \endminipage
213 }

```

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

11.7 Compatibility with hyperref and footnotehyper

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

```

214 \hook_gput_code:nnn { begindocument } { enumext } { \__enumext_after_hyperref: }
215 \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”.

```

216 \cs_new_protected:Nn \__enumext_after_hyperref:
217 {
218     \IfPackageLoadedTF { hyperref }
219     {
220         \bool_set_true:N \l__enumext_hyperref_bool
221         \IfHyperBoolean{hyperfootnotes}
222         {
223             \typeout{hyperfootnotes=true}
224             \bool_set_true:N \l__enumext_footnotes_key_bool
225         }
226         { \typeout{hyperfootnotes=false} }
227     }
228     { }

```

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`.

```

229 \bool_if:NT \l__enumext_footnotes_key_bool
230 {
231     \IfPackageLoadedTF { footnotehyper }
232     {
233         \typeout{OK ~ hyperref ~ and ~ footnotehyper}
234     }
235     {
236         \typeout{No ~ footnotehyper ~ load}
237         \typeout{Load ~ and ~ use ~ \string\makesavenoteenv{enumext*}}
238         \bool_set_false:N \l__enumext_footnotes_key_bool
239     }
240 }

```

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.

```

241 \bool_if:NTF \l__enumext_hyperref_bool
242 {
243     \cs_new_eq:NN \__enumext_hypertarget:nn \hypertarget
244     \cs_new_eq:NN \__enumext_phantomsection: \phantomsection
245 }
246 {
247     \cs_new_eq:NN \__enumext_hypertarget:nn \use_none:nn
248     \cs_new_eq:NN \__enumext_phantomsection: \prg_do_nothing:
249 }
250 }

```

(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.

```

251 \cs_new_protected:Npn \__enumext_newlabel:nn #1 #2
252 {
253   \protected@write \@auxout { }
254   {
255     \token_to_str:N \newlabel {#1}
256     {
257       {#2}
258       \bool_if:NT \l__enumext_hyperref_bool
259       { { \thepage } {#2} {#1} }
260       { }
261     }
262   }
263   \__enumext_hypertarget:nn {#1} { }
264   \__enumext_phantomsection:
265 }

```

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

11.8 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.

```

266 \cs_new_protected:Npn \__enumext_define_counters:Nn #1 #2
267 {
268   \cs_if_exist:cTF { c@ #2 }
269   { \msg_fatal:nnn { enumext } { counters } { #2 } }
270   {
271     \tl_set:Nn #1 { #2 }
272     \newcounter { #2 }
273   }
274 }

```

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

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

```

enumXi      275 \__enumext_define_counters:Nn \l__enumext_counter_i_tl { enumXi }
enumXii     276 \__enumext_define_counters:Nn \l__enumext_counter_ii_tl { enumXii }
enumXiii    277 \__enumext_define_counters:Nn \l__enumext_counter_iii_tl { enumXiii }
enumXvii    278 \__enumext_define_counters:Nn \l__enumext_counter_iv_tl { enumXiv }
enumXviii   279 \__enumext_define_counters:Nn \l__enumext_counter_v_tl { enumXv }
enumXv      280 \__enumext_define_counters:Nn \l__enumext_counter_vi_tl { enumXvi }
enumXvi     281 \__enumext_define_counters:Nn \l__enumext_counter_vii_tl { enumXvii }
enumXviii   282 \__enumext_define_counters:Nn \l__enumext_counter_viii_tl { enumXviii }

```

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

11.9 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.

```

283 \cs_new_protected:Npn \__enumext_register_counter_style:Nn #1 #2
284 {
285   \tl_const:cn { c__enumext_widest_ \cs_to_str:N #1 _tl } {#2}
286   \tl_gput_right:Nn \g__enumext_counter_styles_tl {#1}
287 }
288 \__enumext_register_counter_style:Nn \arabic { 0 }
289 \__enumext_register_counter_style:Nn \Alph { M }
290 \__enumext_register_counter_style:Nn \alph { m }
291 \__enumext_register_counter_style:Nn \Roman { VIII }
292 \__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.

```
293 \cs_new_protected:Npn \__enumext_label_width_by_box:Nn #1#2
294 {
295   \hbox_set:Nn \__enumext_label_width_by_box {#2}
296   \dim_set:Nn #1 { \box_wd:N \__enumext_label_width_by_box }
297 }
298 \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 `\labelstyle` 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.

```
299 \cs_new_protected:Npn \__enumext_label_style:Nnn #1 #2 #3
300 {
301   \tl_clear_new:N #1
302   \tl_put_right:Ne #1 { \tl_trim_spaces:n {#3} }
303   \tl_gset_eq:NN \g__enumext_widest_label_tl #1
304   \tl_map_inline:Nn \g__enumext_counter_styles_tl
305   {
306     \tl_replace_all:Nne #1 { ##1* } { \exp_not:N ##1 {#2} }
307     \tl_greplace_all:Nne \g__enumext_widest_label_tl { ##1* }
308     { \tl_use:c { c__enumext_widest_ \cs_to_str:N ##1 _tl } }
309   }
310   \__enumext_label_width_by_box:Nn \__enumext_current_widest_dim
311   { \tl_use:N \g__enumext_widest_label_tl }
312   \tl_set_eq:cN { the #2 } #1
313 }
314 \cs_generate_variant:Nn \__enumext_label_style:Nnn { cvn }
```

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


11.10 Setting keys associated with label

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

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

```
315 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
316 {
317   \keys_define:nn { enumext / #1 }
318   {
319     font .tl_set:c = { l__enumext_label_font_style_#2_tl },
320     font .value_required:n = true,
321     labelsep .dim_set:c = { l__enumext_labelsep_#2_dim },
322     labelsep .initial:n = {0.3333em},
323     labelsep .value_required:n = true,
324     labelwidth .dim_set:c = { l__enumext_labelwidth_#2_dim },
325     labelwidth .value_required:n = true,
326     wrap-label .cs_set_protected:cp = { __enumext_wrapper_label_#2:n } ##1,
327     wrap-label .initial:n = {##1},
328     wrap-label .value_required:n = true,
329     wrap-label* .code:n = {
330       \bool_set_true:c { l__enumext_wrap_label_opt_#2_bool }
331       \keys_set:nn { enumext / #1 } { wrap-label = {##1} }
332     },
333     wrap-label* .value_required:n = true,
334   }
335 }
336 \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.

```

337 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
338 {
339   \keys_define:nn { enumext / #1 }
340   {
341     align .choice:,
342     align / left .code:n =
343       {
344         \tl_clear:c { l__enumext_label_fill_left_#2_tl }
345         \tl_set:cn { l__enumext_label_fill_right_#2_tl } { \hfill }
346       },
347     align / right .code:n =
348       {
349         \tl_set:cn { l__enumext_label_fill_left_#2_tl } { \hfill }
350         \tl_clear:c { l__enumext_label_fill_right_#2_tl }
351       },
352     align / center .code:n =
353       {
354         \tl_set:cn { l__enumext_label_fill_left_#2_tl } { \hfill }
355         \tl_set:cn { l__enumext_label_fill_right_#2_tl } { \hfill }
356       },
357     align .initial:n = left,
358     align .value_required:n = true,
359   }
360 }
361 \clist_map_inline:nn
362 {
363   {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {keyans}{v}
364 }
365 { \__enumext_tmp:nn #1 }

```

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

```

366 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
367 {
368   \keys_define:nn { enumext / #1 }
369   {
370     align .choice:,
371     align / left .code:n = \str_set:cn { l__enumext_align_label_#2_str } { l },
372     align / right .code:n = \str_set:cn { l__enumext_align_label_#2_str } { r },
373     align / center .code:n = \str_set:cn { l__enumext_align_label_#2_str } { c },
374     align .initial:n = left,
375     align .value_required:n = true,
376   }
377 }
378 \clist_map_inline:nn { {enumext*}{vii}, {keyans*}{viii} } { \__enumext_tmp:nn #1 }

```

(End of definition for `align`.)

11.11 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.

```

379 \cs_new_protected:Nn \__enumext_regex_label_ref_key:
380 {
381   \tl_map_inline:Nn \c__enumext_counter_style_tl
382   {
383     \regex_replace_once:nnN { \c{##1}\* }
384     { \c{##1}\cB{\u{l__enumext_ref_aux_tl}\cE} } \l__enumext_ref_key_arg_tl
385   }
386 }

```

(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`.


```

387 \cs_new_protected:Npn \__enumext_set_label_ref:n #1
388 {
389   \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
390   \tl_set_eq:Nc \l__enumext_ref_aux_tl { \l__enumext_counter_ \l__enumext_level: _tl }
391   \__enumext_regex_label_ref_key:
392   \tl_set_eq:Nc \l__enumext_ref_aux_tl { \l__enumext_the_counter_ \l__enumext_level: _tl }
393   \tl_put_right:ce { \l__enumext_counter_style_for_ref_ \l__enumext_level: _tl }
394   {
395     \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_aux_tl }
396     { \exp_not:V \l__enumext_ref_key_arg_tl }
397   }
398 }

```

(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`.

```

399 \cs_new_protected:Nn \__enumext_use_key_ref:
400 {
401   \tl_if_empty:cF { \l__enumext_counter_style_for_ref_ \l__enumext_level: _tl }
402   {
403     \tl_use:c { \l__enumext_counter_style_for_ref_ \l__enumext_level: _tl }
404   }
405 }

```

(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.

```

406 \cs_new_protected:Npn \__enumext_set_label_ref_h:n #1
407 {
408   \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
409   \int_compare:nNnTF { \l__enumext_level_h_int } = { 1 }
410   {
411     \tl_set_eq:NN \l__enumext_ref_aux_tl \l__enumext_counter_vii_tl
412     \__enumext_regex_label_ref_key:
413     \tl_set_eq:NN \l__enumext_ref_aux_tl \l__enumext_the_counter_vii_tl
414     \tl_put_right:Ne \l__enumext_counter_style_for_ref_vii_tl
415     {
416       \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_aux_tl }
417       { \exp_not:V \l__enumext_ref_key_arg_tl }
418     }
419   }
420   {
421     \tl_set_eq:NN \l__enumext_ref_aux_tl \l__enumext_counter_viii_tl
422     \__enumext_regex_label_ref_key:
423     \tl_set_eq:NN \l__enumext_ref_aux_tl \l__enumext_the_counter_viii_tl
424     \tl_put_right:Ne \l__enumext_counter_style_for_ref_vii_tl
425     {
426       \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_aux_tl }
427       { \exp_not:V \l__enumext_ref_key_arg_tl }
428     }
429   }
430 }

```

(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*`.

```

431 \cs_new_protected:Nn \__enumext_use_key_ref_h:
432 {
433   \int_compare:nNnTF { \l__enumext_level_h_int } = { 1 }
434   {

```

```

435         \tl_if_empty:NF \l__enumext_counter_style_for_ref_vii_tl
436         {
437             \tl_use:N \l__enumext_counter_style_for_ref_vii_tl
438         }
439     }
440 {
441     \tl_if_empty:NF \l__enumext_counter_style_for_ref_viii_tl
442     {
443         \tl_use:N \l__enumext_counter_style_for_ref_viii_tl
444     }
445 }
446 }

```

(End of definition for `__enumext_use_key_ref_h:`.)

11.11.1 Define and set label key for enumext environment

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

```

\__enumext_label_i_tl
\__enumext_label_ii_tl
\__enumext_label_iii_tl
\__enumext_label_iv_tl
447 \cs_set_protected:Npn \__enumext_tmp:nnn #1 #2 #3
448 {
449     \keys_define:nn { enumext / #1 }
450     {
451         label .code:n = {
452             \__enumext_label_style:cvn { \__enumext_label_#2_tl }
453             { \__enumext_counter_#2_tl } {##1}
454             \dim_set_eq:cN { \__enumext_labelwidth_#2_dim }
455             \l__enumext_current_widest_dim
456         },
457         label .initial:n = #3,
458         label .value_required:n = true,
459         ref .code:n = \__enumext_set_label_ref:n {##1},
460         ref .value_required:n = true,
461     }
462 }
463 \__enumext_tmp:nnn { level-1 } { i } { \arabic*. }
464 \__enumext_tmp:nnn { level-2 } { ii } { (\alph*. ) }
465 \__enumext_tmp:nnn { level-3 } { iii } { \roman*. }
466 \__enumext_tmp:nnn { level-4 } { iv } { \Alph*. }

```

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

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

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

```

\__enumext_label_vii_tl
\__enumext_label_viii_tl
467 \cs_set_protected:Npn \__enumext_tmp:nnn #1 #2 #3
468 {
469     \keys_define:nn { enumext / #1 }
470     {
471         label .code:n = {
472             \__enumext_label_style:cvn { \__enumext_label_#2_tl }
473             { \__enumext_counter_#2_tl } {##1}
474             \dim_set_eq:cN { \__enumext_labelwidth_#2_dim }
475             \l__enumext_current_widest_dim
476         },
477         label .initial:n = #3,
478         label .value_required:n = true,
479         ref .code:n = \__enumext_set_label_ref_h:n {##1},
480         ref .value_required:n = true,
481     }
482 }
483 \__enumext_tmp:nnn { enumext* } { vii } { \arabic*. }
484 \__enumext_tmp:nnn { keyans* } { viii } { (\Alph*. ) }

```

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

11.11.3 Define and set label key for keyans and keyanspic environment

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

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

```

485 \keys_define:nn { enumext / keyans }
486 {

```

```

487     label .code:n = {
488         \__enumext_label_style:cvn { l__enumext_label_v_tl }
489         { l__enumext_counter_v_tl } {#1}
490         \dim_set_eq:cN { l__enumext_labelwidth_v_dim }
491         \l__enumext_current_widest_dim
492         \__enumext_label_style:cvn { l__enumext_label_vi_tl }
493         { l__enumext_counter_vi_tl } {#1}
494         \dim_set_eq:cN { l__enumext_labelwidth_v_dim }
495         \l__enumext_current_widest_dim
496     },
497     label .initial:n = (\Alph*),
498     label .value_required:n = true,
499 }

```

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

11.12 Setting start and widest keys

```

\__enumext_start_from:NNn
\__enumext_start_from:ccn

```

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

```

#1: \l__enumext_label_X_tl
#2: \l__enumext_start_X_int
#3: <integer or string>

```

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

```

500 \cs_new_protected:Npn \__enumext_start_from:NNn #1 #2 #3
501 {
502     \__enumext_if_is_int:nTF { #3 }
503     {
504         \int_set:Nn #2 {#3}
505     }
506     {
507         \regex_match:nVT { \c{Alph} | \c{alph} } {#1}
508         { \int_set:Nn #2 { \int_from_alph:n {#3} } }
509         \regex_match:nVT { \c{Roman} | \c{roman} } {#1}
510         { \int_set:Nn #2 { \int_from_roman:n {#3} } }
511     }
512 }
513 \cs_generate_variant:Nn \__enumext_start_from:NNn { ccn }

```

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

```

\__enumext_widest_from:nNNn
\__enumext_widest_from:nccn

```

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

```

#1: The counter associated with the environment level
#2: \l__enumext_label_X_tl
#3: \l__enumext_labelwidth_X_dim
#4: <integer or string>

```

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

```

514 \cs_new_protected:Npn \__enumext_widest_from:nNNn #1 #2 #3 #4
515 {
516     \__enumext_if_is_int:nTF {#4}
517     {
518         \setcounter{enumX#1} { #4 }
519     }
520     {
521         \regex_match:nVT { \c{Alph} | \c{alph} } {#2}
522         { \setcounter{enumX#1} { \int_from_alph:n {#4} } }
523         \regex_match:nVT { \c{Roman} | \c{roman} } {#2}
524         { \setcounter{enumX#1} { \int_from_roman:n {#4} } }
525     }
526     \__enumext_label_width_by_box:cv
527     { l__enumext_labelwidth_#1_dim } { l__enumext_label_#1_tl }
528 }
529 \cs_generate_variant:Nn \__enumext_widest_from:nNNn { nccn }

```

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

```

start
widest
\l__enumext_start_X_int
530 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
531 {
532   \keys_define:nn { enumext / #1 }
533   {
534     start .code:n = {
535       \__enumext_start_from:ccn
536       { \__enumext_label_#2_tl }
537       { \__enumext_start_#2_int } {##1}
538     },
539     start .initial:n = 1,
540     widest .code:n = {
541       \__enumext_widest_from:nccn {#2}
542       { \__enumext_label_#2_tl }
543       { \__enumext_labelwidth_#2_dim } {##1}
544     },
545     widest .value_required:n = true,
546     start .value_required:n = true,
547   }
548 }
549 \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }

```

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

11.13 Setting keys for vertical spaces

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

```

topsep
partopsep
parsep
noitemsep
nosep
550 \cs_set_protected:Npn \__enumext_tmp:nnnnnn #1 #2 #3 #4 #5 #6
551 {
552   \keys_define:nn { enumext / #1 }
553   {
554     topsep .skip_set:c = { \__enumext_topsep_#2_skip },
555     topsep .initial:n = {#3},
556     topsep .value_required:n = true,
557     partopsep .skip_set:c = { \__enumext_partopsep_#2_skip },
558     partopsep .initial:n = {#4},
559     partopsep .value_required:n = true,
560     parsep .skip_set:c = { \__enumext_parsep_#2_skip },
561     parsep .initial:n = {#5},
562     parsep .value_required:n = true,
563     itemsep .skip_set:c = { \__enumext_itemsep_#2_skip },
564     itemsep .initial:n = {#6},
565     itemsep .value_required:n = true,
566     noitemsep .meta:n = { itemsep = 0pt, parsep = 0pt },
567     noitemsep .value_forbidden:n = true,
568     nosep .meta:n = {
569       itemsep = 0pt, parsep = 0pt,
570       topsep = 0pt, partopsep = 0pt,
571     },
572     nosep .value_forbidden:n = true,
573   }
574 }

```

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

```

575 \__enumext_tmp:nnnnnn { level-1 } { i } { 8.0pt plus 2.0pt minus 4.0pt }
576 { 2.0pt plus 1.0pt minus 1.0pt } { 4.0pt plus 2.0pt minus 1.0pt }
577 { 4.0pt plus 2.0pt minus 1.0pt }
578 \__enumext_tmp:nnnnnn { level-2 } { ii } { 4.0pt plus 2.0pt minus 1.0pt }
579 { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
580 { 2.0pt plus 1.0pt minus 1.0pt }
581 \__enumext_tmp:nnnnnn { level-3 } { iii } { 2.0pt plus 1.0pt minus 1.0pt }
582 { 1.0pt minus 1.0pt } { 0pt } { 2.0pt plus 1.0pt minus 1.0pt }
583 \__enumext_tmp:nnnnnn { level-4 } { iv } { 2.0pt plus 1.0pt minus 1.0pt }
584 { 1.0pt minus 1.0pt } { 0pt } { 2.0pt plus 1.0pt minus 1.0pt }
585 \__enumext_tmp:nnnnnn { keyans } { v } { 4.0pt plus 2.0pt minus 1.0pt }
586 { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
587 { 2.0pt plus 1.0pt minus 1.0pt }
588 \__enumext_tmp:nnnnnn { enumext* } { vii } { 8.0pt plus 2.0pt minus 4.0pt }
589 { 2.0pt plus 1.0pt minus 1.0pt } { 4.0pt plus 2.0pt minus 1.0pt }
590 { 4.0pt plus 2.0pt minus 1.0pt }

```

```

591 \__enumext_tmp:nnnnnn { keyans* } { viii } { 4.0pt plus 2.0pt minus 1.0pt }
592 { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
593 { 2.0pt plus 1.0pt minus 1.0pt }

```

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

11.14 Setting keys for horizontal spaces

```

itemindent
rightmargin
listparindent
list-offset
list-indent

```

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

```

594 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
595 {
596   \keys_define:nn { enumext / #1 }
597   {
598     itemindent .dim_set:c = { l__enumext_fake_item_indent_#2_dim },
599     itemindent .value_required:n = true,
600     rightmargin .dim_set:c = { l__enumext_rightmargin_#2_dim },
601     rightmargin .value_required:n = true,
602     listparindent .dim_set:c = { l__enumext_listparindent_#2_dim },
603     listparindent .value_required:n = true,
604     list-offset .dim_set:c = { l__enumext_listoffset_#2_dim },
605     list-offset .value_required:n = true,
606     list-indent .code:n =
607       \bool_set_true:c { l__enumext_leftmargin_tmp_#2_bool }
608       \dim_set:cn { l__enumext_leftmargin_tmp_#2_dim } {##1},
609     list-indent .value_required:n = true,
610   }
611 }
612 \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.

```

613 \cs_set_protected:Npn \__enumext_tmp:n #1
614 {
615   \keys_define:nn { enumext / #1 } { list-indent .initial:n = 0pt, }
616 }
617 \clist_map_inline:nn { enumext*, keyans* } { \__enumext_tmp:n {#1} }

```

11.14.1 Functions for setting the fake `itemindent`

The `itemindent` key does not set the value of `\itemindent`, it only sets the value of the *horizontal space* applied using `\skip_horizontal:N`. We will store this value in the variable and only apply it when it is greater than `0pt`. Here I will need to place `\mode_leave_vertical:` and the plain TeX macro `\ignorespaces` to avoid unwanted extra space when using the `itemindent` key.

```

618 \cs_set_protected:Nn \__enumext_fake_item:
619 {
620   \dim_compare:nNnT
621     { \dim_use:c { l__enumext_fake_item_indent_ \__enumext_level: _dim } }
622     >
623     { \c_zero_dim }
624     {
625       \tl_set:ce { l__enumext_fake_item_indent_ \__enumext_level: _tl }
626       {
627         \exp_not:N \mode_leave_vertical:
628         \exp_not:n { \skip_horizontal:n }
629         { \dim_use:c { l__enumext_fake_item_indent_ \__enumext_level: _dim } }
630         \ignorespaces
631       }
632     }
633 }
634 \cs_set_protected:Nn \__enumext_keyans_fake_item:
635 {
636   \dim_compare:nNnT
637     { \l__enumext_fake_item_indent_v_dim } > { \c_zero_dim }
638     {
639       \tl_set:Ne \l__enumext_fake_item_indent_v_tl
640       {
641         \exp_not:N \mode_leave_vertical:
642         \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_v_dim
643       }
644     }

```

```

645     }
646     \cs_set_protected:Nn \__enumext_fake_item_vii:
647     {
648         \dim_compare:nNnT
649         { \l__enumext_fake_item_indent_vii_dim } > { \c_zero_dim }
650         {
651             \tl_set:Nc \l__enumext_fake_item_indent_vii_tl
652             {
653                 \exp_not:N \mode_leave_vertical:
654                 \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_vii_dim
655             }
656         }
657     }
658     \cs_set_protected:Nn \__enumext_fake_item_viii:
659     {
660         \dim_compare:nNnT
661         { \l__enumext_fake_item_indent_viii_dim } > { \c_zero_dim }
662         {
663             \tl_set:Nc \l__enumext_fake_item_indent_viii_tl
664             {
665                 \exp_not:N \mode_leave_vertical:
666                 \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_viii_dim
667             }
668         }
669     }

```

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

11.15 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.

```

670 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
671 {
672     \keys_define:nn { enumext / #1 }
673     {
674         show-length .bool_set:c = { \l__enumext_show_length_#2_bool },
675         show-length .initial:n = false,
676     }
677 }
678 \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }

```

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

11.16 Setting before, after and first keys

before

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

before*

after

first

```

679 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
680 {
681     \keys_define:nn { enumext / #1 }
682     {
683         before .tl_set:c = { \l__enumext_before_no_starred_key_#2_tl },
684         before .value_required:n = true,
685         before* .tl_set:c = { \l__enumext_before_starred_key_#2_tl },
686         before* .value_required:n = true,
687         after .tl_set:c = { \l__enumext_after_stop_list_#2_tl },
688         after .value_required:n = true,
689         first .tl_set:c = { \l__enumext_after_list_args_#2_tl },
690         first .value_required:n = true,
691     }
692 }
693 \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }

```

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

11.16.1 Functions for before, after and first keys in enumext

`__enumext_before_args_exec:`

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

`__enumext_before_keys_exec:`

`__enumext_after_stop_list:`

`__enumext_after_args_exec:`

```

694 \cs_new_protected:Nn \__enumext_before_args_exec:

```



```

695 {
696   \tl_use:c { l__enumext_before_starred_key_ \__enumext_level: _tl }
697 }

```

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`.

```

698 \cs_new_protected:Nn \__enumext_before_keys_exec:
699 {
700   \tl_use:c { l__enumext_before_no_starred_key_ \__enumext_level: _tl }
701 }

```

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

```

702 \cs_new_protected:Nn \__enumext_after_stop_list:
703 {
704   \tl_use:c { l__enumext_after_stop_list_ \__enumext_level: _tl }
705 }

```

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`.

```

706 \cs_new_protected:Nn \__enumext_after_args_exec:
707 {
708   \tl_use:c { l__enumext_after_list_args_ \__enumext_level: _tl }
709 }

```

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

11.16.2 Functions for before, after and first keys in keyans

```

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

```

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

```

710 \cs_new_protected:Nn \__enumext_before_args_exec_v:
711 {
712   \tl_use:N \l__enumext_before_starred_key_v_tl
713 }

```

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`.

```

714 \cs_new_protected:Nn \__enumext_before_keys_exec_v:
715 {
716   \tl_use:N \l__enumext_before_no_starred_key_v_tl
717 }

```

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

```

718 \cs_new_protected:Nn \__enumext_after_stop_list_v:
719 {
720   \tl_use:N \l__enumext_after_stop_list_v_tl
721 }

```

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`.

```

722 \cs_new_protected:Nn \__enumext_after_args_exec_v:
723 {
724   \tl_use:N \l__enumext_after_list_args_v_tl
725 }

```

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

11.16.3 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.

```

726 \cs_new_protected:Nn \__enumext_before_args_exec_vii:
727 {
728   \tl_use:N \l__enumext_before_starred_key_vii_tl
729 }
730 \cs_new_protected:Nn \__enumext_before_args_exec_viii:
731 {
732   \tl_use:N \l__enumext_before_starred_key_viii_tl
733 }

```

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`.

```

734 \cs_new_protected:Nn \__enumext_before_keys_exec_vii:
735 {
736   \tl_use:N \l__enumext_before_no_starred_key_vii_tl
737 }
738 \cs_new_protected:Nn \__enumext_before_keys_exec_viii:
739 {
740   \tl_use:N \l__enumext_before_no_starred_key_viii_tl
741 }

```

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

```

742 \cs_new_protected:Nn \__enumext_after_stop_list_vii:
743 {
744   \tl_use:N \l__enumext_after_stop_list_vii_tl
745 }
746 \cs_new_protected:Nn \__enumext_after_stop_list_viii:
747 {
748   \tl_use:N \l__enumext_after_stop_list_viii_tl
749 }

```

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`.

```

750 \cs_new_protected:Nn \__enumext_after_args_exec_vii:
751 {
752   \tl_use:N \l__enumext_after_list_args_vii_tl
753 }
754 \cs_new_protected:Nn \__enumext_after_args_exec_viii:
755 {
756   \tl_use:N \l__enumext_after_list_args_viii_tl
757 }

```

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

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

```

758 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
759 {
760   \keys_define:nn { enumext / #1 }
761   {
762     mini-env .dim_set:c = { l__enumext_minipage_right_#2_dim },
763     mini-env .value_required:n = true,
764     mini-sep .dim_set:c = { l__enumext_minipage_hsep_#2_dim },
765     mini-sep .initial:n = 0.3333em,
766     mini-sep .value_required:n = true,
767     columns-sep .dim_set:c = { l__enumext_columns_sep_#2_dim },
768     columns-sep .value_required:n = true,
769     columns .int_set:c = { l__enumext_columns_#2_int },
770     columns .initial:n = 1,
771     columns .value_required:n = true,
772   }
773 }
774 \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`.

```

775 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
776 {
777   \keys_define:nn { enumext / #1 }
778   {
779     columns .initial:n = 2,
780     miniright .tl_gset:c = { g__enumext_miniright_code_#2_tl },
781     miniright .value_required:n = true,
782     miniright* .code:n = {

```

```

783         \bool_gset_true:c { g__enumext_minipage_center_#2_bool }
784         \keys_set:nn { enumext / #1 } { miniright = {##1} }
785     },
786     miniright* .value_required:n = true,
787 }
788 }
789 \clist_map_inline:nn { {enumext*}{vii}, {keyans*}{viii} } { \__enumext_tmp:nn #1 }

```

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

11.18 Adjustment of vertical spaces for multicol

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

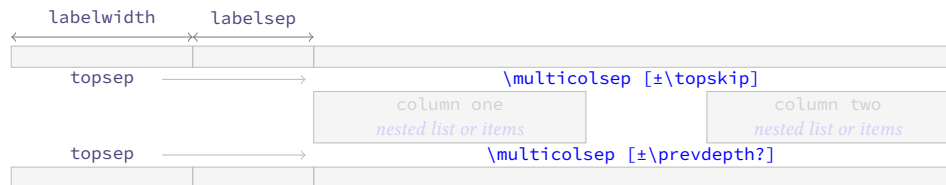


Figure 7: Representation of the vertical space in `multicol` for a nested level.

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

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

11.18.1 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*.

```

790 \cs_new_protected:Nn \__enumext_multi_set_vskip:
791 {
792     \skip_set:cn { \l__enumext_multicols_above_ \__enumext_level: _skip }
793     {
794         \skip_use:c { \l__enumext_topsep_ \__enumext_level: _skip }
795     }
796     \skip_set:cn { \l__enumext_multicols_below_ \__enumext_level: _skip }
797     {
798         \skip_use:c { \l__enumext_topsep_ \__enumext_level: _skip }
799     }
800     \__enumext_add_pre_parsep:
801 }

```

(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*.

```

802 \cs_new_protected:Nn \__enumext_add_pre_parsep:
803 {
804     \int_case:nn { \l__enumext_level_int }
805     {
806         { 2 }{
807             \skip_if_eq:nnF { \l__enumext_parsep_i_skip } { \c_zero_skip }
808             {
809                 \skip_add:Nn \l__enumext_multicols_above_ii_skip { \l__enumext_parsep_i_skip }
810             }
811         }
812         { 3 }{
813             \skip_if_eq:nnF { \l__enumext_parsep_ii_skip } { \c_zero_skip }
814             {

```

```

815             \skip_add:Nn \l__enumext_multicols_above_iii_skip { \l__enumext_parsep_ii_skip
816         }
817     }
818     { 4 }{
819         \skip_if_eq:nnF { \l__enumext_parsep_iii_skip } { \c_zero_skip }
820         {
821             \skip_add:Nn \l__enumext_multicols_above_iv_skip { \l__enumext_parsep_iii_skip
822         }
823     }
824 }
825 }

```

(End of definition for `\l__enumext_add_pre_parsep:`)

`\l__enumext_multi_addvspace:` The function `\l__enumext_multi_addvspace:` will apply the spaces set using `\addvspace` “above” the `multicols` environment in `enumext`, taking into account whether \TeX is in $\langle\textit{horizontal mode}\rangle$ or $\langle\textit{vertical mode}\rangle$.

```

826 \cs_new_protected:Nn \l__enumext_multi_addvspace:
827 {
828     \l__enumext_multi_set_vskip:
829     \mode_if_vertical:T
830     {
831         \skip_add:cn { \l__enumext_multicols_above_ \l__enumext_level: _skip }
832         {
833             \skip_use:c { \l__enumext_partopsep_ \l__enumext_level: _skip }
834         }
835         \skip_add:cn { \l__enumext_multicols_below_ \l__enumext_level: _skip }
836         {
837             \skip_use:c { \l__enumext_partopsep_ \l__enumext_level: _skip }
838         }
839     }
840     \par\nopagebreak
841     \addvspace{ \skip_use:c { \l__enumext_multicols_above_ \l__enumext_level: _skip } }
842 }

```

(End of definition for `\l__enumext_multi_addvspace:`)

11.18.2 Adjustment of vertical spaces for multicols in keyans

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

`\l__enumext_keyans_multi_addvspace:`

```

843 \cs_new_protected:Nn \l__enumext_keyans_multi_set_vskip:
844 {
845     \skip_set:Nn \l__enumext_multicols_above_v_skip
846     {
847         \l__enumext_topsep_v_skip
848     }
849     \skip_set:Nn \l__enumext_multicols_below_v_skip
850     {
851         \l__enumext_topsep_v_skip
852     }
853 }
854 \cs_new_protected:Nn \l__enumext_keyans_multi_addvspace:
855 {
856     \l__enumext_keyans_multi_set_vskip:
857     \mode_if_vertical:T
858     {
859         \skip_add:Nn \l__enumext_multicols_above_v_skip
860         {
861             \skip_use:N \l__enumext_partopsep_v_skip
862         }
863         \skip_add:Nn \l__enumext_multicols_below_v_skip
864         {
865             \skip_use:N \l__enumext_partopsep_v_skip
866         }
867     }
868     \par\nopagebreak
869     \addvspace{ \l__enumext_multicols_above_v_skip }
870 }

```

(End of definition for `\l__enumext_keyans_multi_set_vskip:` and `\l__enumext_keyans_multi_addvspace:`)

11.19 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` 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.

```

871 \DeclareDocumentEnvironment{__enumext_mini_env*}{ m }
872 {
873   \__enumext_minipage:w [ t ] { #1 }
874   \legacy_if_gset_false:n { @minipage }
875   \vspace { 0pt }
876 }
877 { \__enumext_endminipage: }
```

(End of definition for `__enumext_mini_env*`.)

11.19.1 Adjustment of vertical spaces for minipage in enumext

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

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

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

```

878 \cs_new_protected:Nn \__enumext_mini_set_vskip:
879 {
880   \int_compare:nNnTF
881     { \int_use:c { \l__enumext_columns_ \__enumext_level: _int } } > { 1 }
882     {
```

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.

```

883   \skip_if_eq:nnTF
884     { \skip_use:c { \l__enumext_topsep_ \__enumext_level: _skip } } { \c_zero_skip }
885     {
886       \skip_set:Nn \l__enumext_minipage_left_skip
887         {
888           -0.150\box_dp:N \strutbox
889         }
890       \skip_set:Nn \l__enumext_minipage_right_skip
891         {
892           0.695\box_dp:N \strutbox
893         }

```

```

894         \skip_set:Nn \l__enumext_minipage_after_skip
895         {
896             \box_dp:N \strutbox
897         }
898     \__enumext_zero_parsep:
899 }
900 {
901     \skip_set:Nn \l__enumext_minipage_left_skip
902     {
903         \skip_use:c { \l__enumext_topsep_ \__enumext_level: _skip }
904     }
905     \skip_set:Nn \l__enumext_minipage_right_skip
906     {
907         0.695\box_dp:N \strutbox
908     }
909     \skip_set:Nn \l__enumext_minipage_after_skip
910     {
911         1.85\box_dp:N \strutbox
912         + \skip_use:c { \l__enumext_topsep_ \__enumext_level: _skip }
913     }
914 }
915 }
916 {

```

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.

```

917     \skip_if_eq:nnTF
918     { \skip_use:c { \l__enumext_topsep_ \__enumext_level: _skip } } { \c_zero_skip }
919     {
920         \skip_set:Nn \l__enumext_minipage_left_skip
921         {
922             0.5\box_dp:N \strutbox
923             - \skip_use:c { \l__enumext_partopsep_ \__enumext_level: _skip }
924         }
925         \skip_set:Nn \l__enumext_minipage_right_skip
926         {
927             \skip_use:c { \l__enumext_partopsep_ \__enumext_level: _skip }
928         }
929         \skip_set:Nn \l__enumext_minipage_after_skip
930         {
931             1.6\box_dp:N \strutbox
932         }
933     }
934     {
935         \skip_set:Nn \l__enumext_minipage_left_skip
936         {
937             0.5875\box_dp:N \strutbox
938             - \skip_use:c { \l__enumext_partopsep_ \__enumext_level: _skip }
939         }
940         \skip_set:Nn \l__enumext_minipage_right_skip
941         {
942             + \skip_use:c { \l__enumext_topsep_ \__enumext_level: _skip }
943             + \skip_use:c { \l__enumext_partopsep_ \__enumext_level: _skip }
944         }
945         \skip_set:Nn \l__enumext_minipage_after_skip
946         {
947             0.325\box_dp:N \strutbox
948             + \skip_use:c { \l__enumext_topsep_ \__enumext_level: _skip }
949         }
950     }
951 }
952 }

```

(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.

```

953 \cs_new_protected:Nn \__enumext_zero_parsep:
954 {

```



```

955 \int_case:nn { \l__enumext_level_int }
956 {
957   { 2 }{
958     \skip_if_eq:nnF { \l__enumext_parsep_i_skip } { \c_zero_skip }
959     {
960       \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
961     }
962   }
963   { 3 }{
964     \skip_if_eq:nnF { \l__enumext_parsep_ii_skip } { \c_zero_skip }
965     {
966       \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
967     }
968   }
969   { 4 }{
970     \skip_if_eq:nnF { \l__enumext_parsep_iii_skip } { \c_zero_skip }
971     {
972       \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
973     }
974   }
975 }
976 }

```

(End of definition for `__enumext_zero_parsep:`.)

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

```

977 \cs_new_protected:Nn \__enumext_mini_addvspace:
978 {
979   \__enumext_mini_set_vskip:
980   \mode_if_vertical:T
981   {
982     \skip_add:Nn \l__enumext_minipage_left_skip
983     {
984       \skip_use:c { \l__enumext_partopsep_ \l__enumext_level: _skip }
985     }
986     \skip_add:Nn \l__enumext_minipage_after_skip
987     {
988       \skip_use:c { \l__enumext_partopsep_ \l__enumext_level: _skip }
989     }
990   }
991   \par\nopagebreak
992   \addvspace { \l__enumext_minipage_left_skip }
993 }

```

(End of definition for `__enumext_mini_addvspace:`.)

11.19.2 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`.

```

994 \cs_new_protected:Nn \__enumext_keyans_mini_set_vskip:
995 {
996   \skip_zero_new:N \l__enumext_minipage_after_skip
997   \skip_zero_new:N \l__enumext_minipage_left_skip
998   \skip_zero_new:N \l__enumext_minipage_right_skip
999   \int_compare:nNnTF { \l__enumext_columns_v_int } > { 1 }
1000   {
1001     \skip_if_eq:nnTF { \l__enumext_topsep_v_skip } { \c_zero_skip }
1002     {
1003       \skip_set:Nn \l__enumext_minipage_left_skip { -0.25\box_dp:N \strutbox }
1004       \skip_set:Nn \l__enumext_minipage_right_skip { 0.705\box_dp:N \strutbox }
1005       \skip_set:Nn \l__enumext_minipage_after_skip { \box_dp:N \strutbox }
1006       \skip_if_eq:nnF { \l__enumext_parsep_i_skip } { \c_zero_skip }
1007       {
1008         \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
1009       }
1010     }
1011   }

```

```

1011     {
1012         \skip_set:Nn \l__enumext_minipage_left_skip
1013         {
1014             \skip_use:N \l__enumext_topsep_v_skip
1015         }
1016         \skip_set:Nn \l__enumext_minipage_right_skip
1017         {
1018             0.705\box_dp:N \strutbox
1019         }
1020         \skip_set:Nn \l__enumext_minipage_after_skip
1021         {
1022             1.85\box_dp:N \strutbox + \l__enumext_topsep_v_skip
1023         }
1024     }
1025 }
1026 {
1027     \skip_if_eq:nnTF { \l__enumext_topsep_v_skip } { \c_zero_skip }
1028     {
1029         \skip_set:Nn \l__enumext_minipage_left_skip
1030         {
1031             0.5\box_dp:N \strutbox
1032             + \l__enumext_partopsep_v_skip
1033         }
1034         \skip_set:Nn \l__enumext_minipage_right_skip
1035         {
1036             \l__enumext_partopsep_v_skip
1037         }
1038         \skip_set:Nn \l__enumext_minipage_after_skip { 1.6\box_dp:N \strutbox }
1039     }
1040     {
1041         \skip_set:Nn \l__enumext_minipage_left_skip
1042         {
1043             0.5875\box_dp:N \strutbox - \l__enumext_partopsep_v_skip
1044         }
1045         \skip_set:Nn \l__enumext_minipage_right_skip
1046         {
1047             \l__enumext_topsep_v_skip + \l__enumext_partopsep_v_skip
1048         }
1049         \skip_set:Nn \l__enumext_minipage_after_skip
1050         {
1051             0.325\box_dp:N \strutbox + \l__enumext_topsep_v_skip
1052         }
1053     }
1054 }
1055 }

```

(End of definition for `__enumext_keyans_mini_set_vskip:`)

`__enumext_keyans_mini_addvspace:`

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

```

1056 \cs_new_protected:Nn \__enumext_keyans_mini_addvspace:
1057 {
1058     \__enumext_keyans_mini_set_vskip:
1059     \mode_if_vertical:T
1060     {
1061         \skip_add:Nn \l__enumext_minipage_left_skip
1062         {
1063             \l__enumext_partopsep_v_skip
1064         }
1065         \skip_add:Nn \l__enumext_minipage_after_skip
1066         {
1067             \l__enumext_partopsep_v_skip
1068         }
1069     }
1070     \par\nopagebreak
1071     \addvspace { \l__enumext_minipage_left_skip }
1072 }

```

(End of definition for `__enumext_keyans_mini_addvspace:`)

11.19.3 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*`.

```

1073 \cs_new_protected:Nn \__enumext_mini_set_vskip_vii:
1074 {
1075   \skip_zero_new:N \l__enumext_minipage_left_skip
1076   \skip_gzero_new:N \g__enumext_minipage_right_skip
1077   \skip_gzero_new:N \g__enumext_minipage_after_skip
1078   \skip_if_eq:nnTF { \l__enumext_topsep_vii_skip } { \c_zero_skip }
1079   {
1080     \skip_set:Nn \l__enumext_minipage_left_skip { 0.5\box_dp:N \strutbox }
1081     \skip_gset:Nn \g__enumext_minipage_right_skip { 0.325\box_dp:N \strutbox }
1082   }
1083   {
1084     \skip_set:Nn \l__enumext_minipage_left_skip { 0.5875\box_dp:N \strutbox }
1085     \skip_gset:Nn \g__enumext_minipage_right_skip
1086     {
1087       \l__enumext_topsep_vii_skip
1088     }
1089     \skip_gset:Nn \g__enumext_minipage_after_skip
1090     {
1091       0.325\box_dp:N \strutbox + \l__enumext_topsep_vii_skip
1092     }
1093   }
1094 }
1095 \cs_new_protected:Nn \__enumext_mini_set_vskip_viii:
1096 {
1097   \skip_zero_new:N \l__enumext_minipage_after_skip
1098   \skip_zero_new:N \l__enumext_minipage_left_skip
1099   \skip_zero_new:N \l__enumext_minipage_right_skip
1100   \skip_if_eq:nnTF { \l__enumext_topsep_viii_skip } { \c_zero_skip }
1101   {
1102     \skip_set:Nn \l__enumext_minipage_left_skip
1103     {
1104       0.5\box_dp:N \strutbox
1105     }
1106     \skip_set:Nn \l__enumext_minipage_right_skip
1107     {
1108       \l__enumext_partopsep_viii_skip
1109     }
1110     \skip_set:Nn \l__enumext_minipage_after_skip
1111     {
1112       1.6\box_dp:N \strutbox
1113     }
1114   }
1115   {
1116     \skip_set:Nn \l__enumext_minipage_left_skip
1117     {
1118       0.5875\box_dp:N \strutbox
1119     }
1120     \skip_set:Nn \l__enumext_minipage_right_skip
1121     {
1122       \l__enumext_topsep_viii_skip
1123     }
1124     \skip_set:Nn \l__enumext_minipage_after_skip
1125     {
1126       0.325\box_dp:N \strutbox + \l__enumext_topsep_viii_skip
1127     }
1128   }
1129 }

```

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

`__enumext_mini_addvspace_vii:`
`__enumext_mini_addvspace_viii:`

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

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

```

1130 \cs_new_protected:Nn \__enumext_mini_addvspace_vii:

```

```

1131 {
1132   \__enumext_mini_set_vskip_vii:
1133   \par\nopagebreak
1134   \addvspace { \l__enumext_minipage_left_skip }
1135 }
1136 \cs_new_protected:Nn \__enumext_mini_addvspace_viii:
1137 {
1138   \__enumext_mini_set_vskip_viii:
1139   \par\nopagebreak
1140   \addvspace { \l__enumext_minipage_left_skip }
1141 }

```

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

11.19.4 The command \miniright

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

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

```

1142 \NewDocumentCommand \miniright { s }
1143 {
1144   \int_compare:nNnT { \l__enumext_keyans_pic_level_int } = { 1 }
1145   {
1146     \msg_error:nnn { enumext } { wrong-miniright-place }
1147   }
1148   \int_compare:nNnT { \l__enumext_level_int } = { 0 }
1149   {
1150     \msg_error:nnn { enumext } { wrong-miniright-place }
1151   }
1152   \int_compare:nNnTF { \l__enumext_keyans_level_int } = { 1 }
1153   {
1154     \__enumext_keyans_mini_right_cmd:n {#1}
1155   }
1156   { \__enumext_mini_right_cmd:n {#1} }
1157 }

```

(End of definition for \miniright. This function is documented on page 8.)

`__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 `\l__enumext_minipage_right_X_dim`, if so we close the `\multicols` environment with the `__enumext_mini_env*` environment on the “left side”, then we open the `__enumext_mini_env*` environment on the “right side”, apply our adjusted “vertical spaces”, followed by adding the `\centering` command when the starred argument ‘*’ is not present and set zero `\g__enumext_minipage_stat_int`, otherwise we return an error.

```

1158 \cs_new_protected:Npn \__enumext_mini_right_cmd:n #1
1159 {
1160   \dim_compare:nNnTF
1161   { \dim_use:c { \l__enumext_minipage_right_ \__enumext_level: _dim } } > { \c_zero_dim }
1162   {
1163     \__enumext_multicols_stop:
1164     \end{\__enumext_mini_env*}
1165     \hfill
1166     \begin{\__enumext_mini_env*}
1167     { \dim_use:c { \l__enumext_minipage_right_ \__enumext_level: _dim } }
1168     \par\addvspace { \l__enumext_minipage_right_skip }
1169     \bool_if:nF {#1}
1170     {
1171       \centering
1172     }
1173     \int_gzero:N \g__enumext_minipage_stat_int
1174   }
1175   { \msg_error:nnn { enumext } { wrong-miniright-use } }
1176 }

```

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

`__enumext_keyans_mini_right_cmd:n`

The function `__enumext_keyans_mini_right_cmd:n` takes as argument the *starred version* ‘`*`’ of the `\mini_right` command in the `keyans` environment. The implementation of this function is the same as that of the `__enumext_mini_right_cmd:n` function of the `enumext` environment.

```

1177 \cs_new_protected:Npn \__enumext_keyans_mini_right_cmd:n #1
1178 {
1179   \dim_compare:nNnTF { \l__enumext_minipage_right_v_dim } > { \c_zero_dim }
1180   {
1181     \__enumext_keyans_multicols_stop:
1182     \end{__enumext_mini_env*}
1183     \hfill
1184     \begin{__enumext_mini_env*}{ \l__enumext_minipage_right_v_dim }
1185     \par\addvspace { \l__enumext_minipage_right_skip }
1186     \bool_if:nF {#1}
1187     {
1188       \centering
1189     }
1190     \int_gzero:N \g__enumext_minipage_stat_int
1191   }
1192   { \msg_error:nnn { enumext } { wrong-miniright-use } }
1193 }

```

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

11.20 Setting above and below keys

While having controlled the *vertical spaces* within the `enumext` and `keyans` environments when using the `columns` or `mini-env` keys, sometimes the “vertical spaces above” or “vertical spaces below” the environments are not as expected and it is necessary to be able to apply a “fine correction” to these. As I have not been able to correct these *glitches*, the best option is to leave a couple of *keys* dedicated to this purpose, in this case it is best to use `\vspace` or `\vspace*` when convenient.

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

`above`
`above*`
`below`
`below*`

```

1194 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1195 {
1196   \keys_define:nn { enumext / #1 }
1197   {
1198     above .skip_set:c = { \l__enumext_vspace_above_#2_skip },
1199     above .value_required:n = true,
1200     above* .code:n      = \bool_set_true:c { \l__enumext_vspace_a_star_#2_bool }
1201                      \keys_set:nn { enumext / #1 } { above = {##1} },
1202     above* .value_required:n = true,
1203     below .skip_set:c = { \l__enumext_vspace_below_#2_skip },
1204     below .value_required:n = true,
1205     below* .code:n      = \bool_set_true:c { \l__enumext_vspace_b_star_#2_bool }
1206                      \keys_set:nn { enumext / #1 } { below = {##1} },
1207     below* .value_required:n = true,
1208   }
1209 }
1210 \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }

```

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

11.20.1 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.

```

1211 \cs_new_protected:Nn \__enumext_vspace_above:
1212 {
1213   \skip_if_eq:nnF
1214   { \skip_use:c { \l__enumext_vspace_above_ \__enumext_level: _skip } } { \c_zero_skip }
1215   {
1216     \bool_if:cTF { \l__enumext_vspace_a_star_ \__enumext_level: _bool }
1217     {
1218       \vspace*{ \skip_use:c { \l__enumext_vspace_above_ \__enumext_level: _skip } }
1219     }
1220     {
1221       \vspace { \skip_use:c { \l__enumext_vspace_above_ \__enumext_level: _skip } }
1222     }
1223   }
1224 }

```

(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.

```

1225 \cs_new_protected:Nn \__enumext_vspace_below:
1226 {
1227   \skip_if_eq:nnF
1228   { \skip_use:c { \__enumext_vspace_below_ \__enumext_level: _skip } } { \c_zero_skip }
1229   {
1230     \bool_if:cTF { \__enumext_vspace_b_star_ \__enumext_level: _bool }
1231     {
1232       \vspace*{ \skip_use:c { \__enumext_vspace_below_ \__enumext_level: _skip } }
1233     }
1234     {
1235       \vspace { \skip_use:c { \__enumext_vspace_below_ \__enumext_level: _skip } }
1236     }
1237   }
1238 }

```

(End of definition for `__enumext_vspace_below:`.)

11.20.2 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.

```

1239 \cs_new_protected:Nn \__enumext_vspace_above_v:
1240 {
1241   \skip_if_eq:nnF { \l__enumext_vspace_above_v_skip } { \c_zero_skip }
1242   {
1243     \bool_if:NTF \l__enumext_vspace_a_star_v_bool
1244     {
1245       \vspace*{ \l__enumext_vspace_above_v_skip }
1246     }
1247     { \vspace { \l__enumext_vspace_above_v_skip } }
1248   }
1249 }

```

(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.

```

1250 \cs_new_protected:Nn \__enumext_vspace_below_v:
1251 {
1252   \skip_if_eq:nnF { \l__enumext_vspace_below_v_skip } { \c_zero_skip }
1253   {
1254     \bool_if:NTF \l__enumext_vspace_b_star_v_bool
1255     {
1256       \vspace*{ \l__enumext_vspace_below_v_skip }
1257     }
1258     { \vspace { \l__enumext_vspace_below_v_skip } }
1259   }
1260 }

```

(End of definition for `__enumext_vspace_below_v:`.)

11.20.3 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:`

```

1261 \cs_new_protected:Nn \__enumext_vspace_above_vii:
1262 {
1263   \skip_if_eq:nnF { \l__enumext_vspace_above_vii_skip } { \c_zero_skip }
1264   {
1265     \bool_if:NTF \l__enumext_vspace_a_star_vii_bool
1266     {
1267       \vspace*{ \l__enumext_vspace_above_vii_skip }
1268     }
1269     { \vspace { \l__enumext_vspace_above_vii_skip } }
1270   }
1271 }
1272 \cs_new_protected:Nn \__enumext_vspace_above_viii:
1273 {
1274   \skip_if_eq:nnF { \l__enumext_vspace_above_viii_skip } { \c_zero_skip }

```



```

1275     {
1276         \bool_if:NTF \l__enumext_vspace_a_star_viii_bool
1277         {
1278             \vspace*{ \l__enumext_vspace_above_viii_skip }
1279         }
1280         { \vspace { \l__enumext_vspace_above_viii_skip } }
1281     }
1282 }

```

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

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

```

1283 \cs_new_protected:Nn \l__enumext_vspace_below_vii:
1284 {
1285     \skip_if_eq:nnF { \l__enumext_vspace_below_vii_skip } { \c_zero_skip }
1286     {
1287         \bool_if:NTF \l__enumext_vspace_b_star_vii_bool
1288         {
1289             \vspace*{ \l__enumext_vspace_below_vii_skip }
1290         }
1291         { \vspace { \l__enumext_vspace_below_vii_skip } }
1292     }
1293 }
1294 \cs_new_protected:Nn \l__enumext_vspace_below_viii:
1295 {
1296     \skip_if_eq:nnF { \l__enumext_vspace_below_viii_skip } { \c_zero_skip }
1297     {
1298         \bool_if:NTF \l__enumext_vspace_b_star_viii_bool
1299         {
1300             \vspace*{ \l__enumext_vspace_below_viii_skip }
1301         }
1302         { \vspace { \l__enumext_vspace_below_viii_skip } }
1303     }
1304 }

```

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

11.21 Setting save-ans and resume keys

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

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

```

save-ans \keys_define:nn { enumext / level-1 }
resume  {
resume*  {
1305     \keys_define:nn { enumext / level-1 }
1306     {
1307         save-ans .code:n = \__enumext_storing_set:n {#1},
1308         save-ans .value_required:n = true,
1309         resume  .code:n = \__enumext_resume_counter:,
1310         resume  .value_forbidden:n = true,
1311         resume* .code:n = \__enumext_resume_counter_star:,
1312         resume* .value_forbidden:n = true,
1313     }
1314     \keys_define:nn { enumext / enumext* }
1315     {
1316         save-ans .code:n = \__enumext_storing_set:n {#1},
1317         save-ans .value_required:n = true,
1318         resume  .code:n = \__enumext_resume_counter_vii:,
1319         resume  .value_forbidden:n = true,
1320     }

```

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

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

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

```

1321 \cs_new_protected:Npn \__enumext_storing_set:n #1
1322 {
1323   \tl_set:Nx \l__enumext_store_name_tl {#1}
1324   \bool_set_true:N \l__enumext_store_active_bool
1325   \int_if_exist:cF { g__enumext_resume_#1_int }
1326   {
1327     \int_new:c { g__enumext_resume_#1_int }
1328   }
1329   \bool_if:NT \l__enumext_check_ans_bool
1330   {
1331     \__enumext_check_ans_int:n {#1}
1332   }
1333 }

```

(End of definition for __enumext_storing_set:n.)

```

\__enumext_resume_counter:
\__enumext_resume_counter_vii:

```

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

```

1334 \cs_new_protected:Nn \__enumext_resume_counter:
1335 {
1336   \bool_if:NT \l__enumext_store_active_bool
1337   {
1338     \int_gset:Nn \g__enumext_resume_int
1339     {
1340       \int_use:c { g__enumext_resume_ \l__enumext_store_name_tl _int }
1341     }
1342   }
1343   \bool_set_true:N \l__enumext_resume_bool
1344 }
1345 \cs_new_protected:Nn \__enumext_resume_counter_vii:
1346 {
1347   \bool_if:NT \l__enumext_store_active_bool
1348   {
1349     \int_gset:Nn \g__enumext_resume_int
1350     {
1351       \int_use:c { g__enumext_resume_ \l__enumext_store_name_tl _int }
1352     }
1353   }
1354   \bool_set_true:N \l__enumext_resume_vii_bool
1355 }

```

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

11.22 Setting check-ans key

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

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

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

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

Each \item and \item* in the environment must be matched with the counter associated with \anskey (\g__enumext_count_item_ans_int). We analyze the cases:

- a) If the list only has one level the number of \item + \item* = \anskey
- b) If the list has *nested levels*, for each level of nesting we need to increase by one (for the \item or \item* that opens the nest) so that the account remains the same.
- c) If there is the option *no-store* we must add the items within this level plus one to maintain the equality.

With keyans, keyans* and keyanspic it is enough to increase in one the integer of \anskey. The integers created must be global if they are not lost in the interior levels of nesting and to execute the test we will use a “hook” function after closing the first level of the environment.

11.22.1 The check answer mechanism

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

```

check-ans \cs_set_protected:Npn \__enumext_tmp:n #1
no-store {
1356   \keys_define:nn { enumext / #1 }
1357   {
1358     {
1359       check-ans .bool_set:N = \__enumext_check_ans_bool,
1360       check-ans .initial:n = false,
1361       no-store .bool_set:N = \__enumext_store_ans_bool,
1362       no-store .initial:n = false,
1363     }
1364   }
1365 }
1366 \clist_map_inline:nn
1367 {
1368   level-1, level-2, level-3, level-4, enumext*
1369 }
1370 { \__enumext_tmp:n {#1} }
```

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

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

```

1371 \cs_new_protected:Npn \__enumext_check_ans_int:n #1
1372 {
1373   \int_if_exist:cF { g__enumext_count_#1_item_ans_int }
1374   { \int_new:c { g__enumext_count_#1_item_ans_int } }
1375   \int_if_exist:cF { g__enumext_count_#1_i_int }
1376   { \int_new:c { g__enumext_count_#1_i_int } }
1377   \int_if_exist:cF { g__enumext_count_#1_ii_int }
1378   { \int_new:c { g__enumext_count_#1_ii_int } }
1379   \int_if_exist:cF { g__enumext_count_#1_iii_int }
1380   { \int_new:c { g__enumext_count_#1_iii_int } }
1381   \int_if_exist:cF { g__enumext_count_#1_iv_int }
1382   { \int_new:c { g__enumext_count_#1_iv_int } }
1383   \int_if_exist:cF { g__enumext_count_#1_vii_int }
1384   { \int_new:c { g__enumext_count_#1_vii_int } }
```

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

```

1385   \bool_lazy_all:nTF
1386   {
1387     { \g__enumext_starred_bool }
1388     { \int_compare_p:nNn { \__enumext_level_int } = { \c_zero_int } }
1389   }
1390   {
1391     \int_gset_eq:Nc \g__enumext_count_item_all_int { g__enumext_count_#1_vii_int }
1392   }
1393   {
1394     \int_gset_eq:Nc \g__enumext_count_item_all_int { g__enumext_count_#1_i_int }
1395   }
1396   \int_gset_eq:Nc \g__enumext_count_item_i_int { g__enumext_count_#1_i_int }
1397   \int_gset_eq:Nc \g__enumext_count_item_ii_int { g__enumext_count_#1_ii_int }
1398   \int_gset_eq:Nc \g__enumext_count_item_iii_int { g__enumext_count_#1_iii_int }
1399   \int_gset_eq:Nc \g__enumext_count_item_iv_int { g__enumext_count_#1_iv_int }
1400   \int_gset_eq:Nc \g__enumext_count_item_vii_int { g__enumext_count_#1_vii_int }
1401   \int_gset_eq:Nc \g__enumext_count_item_ans_int { g__enumext_count_#1_item_ans_int }
1402 }
```

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

11.22.2 Set-up check answer mechanism

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

```

1403 \cs_new_protected:Nn \__enumext_check_ans_count:
1404 {
1405   \bool_if:NT \l__enumext_check_ans_bool
```

```

1406     {
1407         \bool_if:NTF \l__enumext_store_ans_bool
1408         {
1409             \int_gadd:cn { g__enumext_count_item_ \__enumext_level: _int }
1410             { \int_use:c { g__enumext_count_level_ \__enumext_level: _int } + 1 }
1411         }
1412         { \int_gincr:c { g__enumext_count_item_ \__enumext_level: _int } }
1413     }
1414 }

```

(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:

```

1415 \cs_new_protected:Nn \__enumext_check_ans_active:
1416 {
1417     \int_set:Nn \l__enumext_compare_items_ans_int
1418     {
1419         \g__enumext_count_item_all_int - \g__enumext_count_item_ii_int
1420         - \g__enumext_count_item_iii_int - \g__enumext_count_item_iv_int
1421     }
1422     \int_compare:nNnTF
1423     { \l__enumext_compare_items_ans_int } = { \g__enumext_count_item_ans_int }
1424     {
1425         \msg_term:nnV { enumext } { items-same-answer }
1426         \g__enumext_store_name_tl
1427     }
1428     {
1429         \msg_warning:nnV { enumext } { item-different-answer }
1430         \g__enumext_store_name_tl
1431     }

```

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

```

1432     \int_gzero:N \g__enumext_count_level_i_int
1433     \int_gzero:N \g__enumext_count_level_ii_int
1434     \int_gzero:N \g__enumext_count_level_iii_int
1435     \int_gzero:N \g__enumext_count_level_iv_int
1436     \int_gzero:N \g__enumext_count_level_vii_int
1437 }

```

```

1438 \cs_new_protected:Nn \__enumext_check_ans_active_vii:
1439 {
1440     \int_set:Nn \l__enumext_compare_items_ans_int
1441     {
1442         \g__enumext_count_item_all_int
1443         - \g__enumext_count_item_i_int
1444         - \g__enumext_count_item_ii_int
1445         - \g__enumext_count_item_iii_int
1446         - \g__enumext_count_item_iv_int
1447     }
1448     \int_compare:nNnTF
1449     { \l__enumext_compare_items_ans_int } = { \g__enumext_count_item_ans_int }
1450     {
1451         \msg_term:nnV { enumext } { items-same-answer }
1452         \g__enumext_store_name_tl
1453     }
1454     {
1455         \msg_warning:nnV { enumext } { item-different-answer }
1456         \g__enumext_store_name_tl
1457     }
1458     \int_gzero:N \g__enumext_count_level_i_int
1459     \int_gzero:N \g__enumext_count_level_ii_int
1460     \int_gzero:N \g__enumext_count_level_iii_int
1461     \int_gzero:N \g__enumext_count_level_iv_int
1462     \int_gzero:N \g__enumext_count_level_vii_int
1463 }

```

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

11.23 Keys and functions associated with storage

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

```

wrap-ans 1464 \cs_set_protected:Npn \__enumext_tmp:n #1
mark-ans 1465 {
mark-pos 1466   \keys_define:nn { enumext / #1 }
store-ref 1467   {
1468     wrap-ans .cs_set_protected:Np = \__enumext_anskey_wrapper:n ##1,
1469     wrap-ans .initial:n = \fbox{##1},
1470     wrap-ans .value_required:n = true,
1471     mark-ans .code:n = \tl_set:Nn \__enumext_mark_answer_sym_tl {##1},
1472     mark-ans .initial:n = \textasteriskcentered,
1473     mark-ans .value_required:n = true,
1474     mark-pos .choice:,
1475     mark-pos / left .code:n = \str_set:Nn \__enumext_mark_position_str { l },
1476     mark-pos / right .code:n = \str_set:Nn \__enumext_mark_position_str { r },
1477     mark-pos .initial:n = right,
1478     mark-pos .value_required:n = true,
1479     show-ans .code:n = \bool_set_true:N \__enumext_show_answer_bool
1480               \bool_set_false:N \__enumext_show_position_bool,
1481     show-ans .value_forbidden:n = true,
1482     show-pos .code:n = \bool_set_true:N \__enumext_show_position_bool
1483               \bool_set_false:N \__enumext_show_answer_bool,
1484     show-pos .value_forbidden:n = true,
1485     mark-ref .code:n = \tl_set:Nn \__enumext_mark_ref_sym_tl {##1},
1486     mark-ref .initial:n = \textasteriskcentered,
1487     mark-ref .value_required:n = true,
1488     store-ref .bool_set:N = \__enumext_store_ref_key_bool,
1489     store-ref .initial:n = false,
1490   }
1491 }
1492 \clist_map_inline:nn { level-1, enumext* } { \__enumext_tmp:n {#1} }

```

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

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

```

mark-pos 1493 \cs_set_protected:Npn \__enumext_tmp:n #1
show-ans 1494 {
1495   \keys_define:nn { enumext / #1 }
1496   {
1497     mark-pos .choice:,
1498     mark-pos / left .code:n = \str_set:Nn \__enumext_mark_position_str { l },
1499     mark-pos / right .code:n = \str_set:Nn \__enumext_mark_position_str { r },
1500     mark-pos .initial:n = right,
1501     mark-pos .value_required:n = true,
1502     show-ans .code:n = \bool_set_true:N \__enumext_show_answer_bool
1503                       \bool_set_false:N \__enumext_show_position_bool,
1504     show-ans .value_forbidden:n = true,
1505     show-pos .code:n = \bool_set_true:N \__enumext_show_position_bool
1506                       \bool_set_false:N \__enumext_show_answer_bool,
1507     show-pos .value_forbidden:n = true,
1508   }
1509 }
1510 \clist_map_inline:nn { keyans, keyans* } { \__enumext_tmp:n {#1} }

```

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

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

```

1511 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1512 {
1513   \keys_define:nn { enumext / #1 }
1514   {
1515     columns* .code:n = \bool_set_true:c { l__enumext_store_columns_#2_bool }
1516               \int_set:cn { l__enumext_store_columns_#2_int } {##1}
1517               \tl_put_right:ce { l__enumext_store_opt_#2_tl }

```

```

1518         {
1519             columns = \exp_not:v { l__enumext_store_columns_#2_int },
1520         },,
1521         columns* .value_required:n = true,
1522         columns-sep* .code:n = \bool_set_true:c { l__enumext_store_columns_sep_#2_bool }
1523             \dim_set:cn { l__enumext_store_columns_sep_#2_dim } {##1}
1524             \tl_put_right:ce { l__enumext_store_opt_#2_tl }
1525             {
1526                 columns-sep = \exp_not:v { l__enumext_store_columns_sep_#2_dim },
1527             },
1528         columns-sep* .value_required:n = true,
1529     }
1530 }
1531 \clist_map_inline:nn
1532 {
1533     {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {enumext*}{vii}
1534 }
1535 { \__enumext_tmp:nn #1 }

```

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

11.23.1 Function for storing content in prop list

```

\__enumext_store_addto_prop:n
\__enumext_store_addto_prop:V

```

The function `__enumext_store_addto_prop:n` stores the content in *⟨prop list⟩* defined by `save-ans` key, if it does not exist it will create it globally. The “*stored content*” is retrieved by means of the `\getkeyans` command.

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

```

1536 \cs_new_protected:Npn \__enumext_store_addto_prop:n #1
1537 {
1538     \prop_if_exist:cF { g__enumext_ \l__enumext_store_name_tl _prop }
1539     { \prop_new:c { g__enumext_ \l__enumext_store_name_tl _prop } }
1540     \prop_gput:cen { g__enumext_ \l__enumext_store_name_tl _prop }
1541     {
1542         \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } +1 }
1543     } { #1 }
1544 }
1545 \cs_generate_variant:Nn \__enumext_store_addto_prop:n { V }

```

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

11.23.2 Function for storing content in sequence

```

\__enumext_store_addto_seq:n
\__enumext_store_addto_seq:v
\__enumext_store_addto_seq:V

```

The function `__enumext_store_addto_seq:n` stores the content in *⟨sequence⟩* defined by `save-ans` key, if it does not exist it will create it globally. This function is used by `\anskey` in `enumext`, `\item*` in `keyans` and `\anspic` in `keyanspic`. The form in which the content is stored in *⟨sequence⟩* is in a internal `enumext` or `enumext*` environments with the *same structure* in which the command was executed.

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

```

1546 \cs_new_protected:Npn \__enumext_store_addto_seq:n #1
1547 {
1548     \seq_if_exist:cF { g__enumext_ \l__enumext_store_name_tl _seq }
1549     { \seq_new:c { g__enumext_ \l__enumext_store_name_tl _seq } }
1550     \seq_gput_right:cn { g__enumext_ \l__enumext_store_name_tl _seq } { #1 }
1551 }
1552 \cs_generate_variant:Nn \__enumext_store_addto_seq:n { v, V }

```

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

11.23.3 Functions for storing the list structure in the sequence

```

\__enumext_store_level_open:
\__enumext_store_level_close:

```

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

```

1553 \cs_new_protected:Nn \__enumext_store_level_open:
1554 {
1555     \bool_if:NF \l__enumext_store_ans_bool
1556     {

```



```

1557     \tl_if_empty:cTF { l__enumext_store_opt_ \__enumext_level: _tl }
1558     {
1559         \__enumext_store_addto_seq:n
1560         {
1561             \item \begin{enumext}
1562         }
1563     }
1564     {
1565         \tl_put_left:cn { l__enumext_store_opt_ \__enumext_level: _tl }
1566         {
1567             \item \begin{enumext} [
1568         }
1569         \tl_put_right:cn { l__enumext_store_opt_ \__enumext_level: _tl }
1570         {
1571             ]
1572         }
1573         \__enumext_store_addto_seq:v { l__enumext_store_opt_ \__enumext_level: _tl }
1574     }
1575 }
1576 }
1577 \cs_new_protected:Nn \__enumext_store_level_close:
1578 {
1579     \bool_if:NF \l__enumext_store_ans_bool
1580     {
1581         \__enumext_store_addto_seq:n { \end{enumext} }
1582     }
1583 }

```

(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.

```

1584 \cs_new_protected:Nn \__enumext_store_level_open_vii:
1585 {
1586     \bool_if:NF \l__enumext_store_ans_bool
1587     {
1588         \tl_if_empty:NTF \l__enumext_store_opt_vii_tl
1589         {
1590             \__enumext_store_addto_seq:n
1591             {
1592                 \item \mode_leave_vertical:
1593                 \vspace { -\skip_eval:n { \baselineskip + \parsep } }
1594                 \begin{enumext*}[before={\setlength{\topsep}{\opt}},]
1595             }
1596         }
1597         {
1598             \tl_put_left:Nn \l__enumext_store_opt_vii_tl
1599             {
1600                 \item \mode_leave_vertical:
1601                 \vspace { -\skip_eval:n { \baselineskip + \parsep } }
1602                 \begin{enumext*}[before={\setlength{\topsep}{\opt}},
1603             }
1604             \tl_put_right:Nn \l__enumext_store_opt_vii_tl
1605             {
1606                 ]
1607             }
1608             \__enumext_store_addto_seq:v \l__enumext_store_opt_vii_tl
1609         }
1610     }
1611 }
1612 \cs_new_protected:Nn \__enumext_store_level_close_vii:
1613 {
1614     \bool_if:NF \l__enumext_store_ans_bool
1615     {
1616         \__enumext_store_addto_seq:n { \end{enumext*} }
1617     }
1618 }

```

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

11.23.4 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`

```
1619 \cs_new_protected:Nn \__enumext_print_keyans_box:NN
1620 {
1621   \mode_leave_vertical:
1622   \skip_horizontal:n { -\dim_use:N #2 }
1623   \makebox[0pt][ r ]
1624   {
1625     \makebox[ \dim_use:N #1 ][ \l__enumext_mark_position_str ]
1626     {
1627       \tl_use:N \l__enumext_mark_answer_sym_tl
1628     }
1629   }
1630   \skip_horizontal:n { \dim_use:N #2 }
1631 }
1632 \cs_generate_variant:Nn \__enumext_print_keyans_box:NN { cc }
```

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

11.24 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`.

```
1633 \keys_define:nn { enumext / anskey }
1634 {
1635   item-sym* .tl_set:N = \l__enumext_store_item_symbol_tl,
1636   item-sym* .value_required:n = true,
1637   item-pos* .dim_set:N = \l__enumext_store_item_symbol_sep_dim,
1638   item-pos* .value_required:n = true,
1639   store-brk .bool_set:N = \l__enumext_store_columns_break_bool,
1640   store-brk .default:n = true,
1641   store-brk .value_forbidden:n = true,
1642 }
```

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

```
1643 \NewDocumentCommand \anskey { d() s o +m }
1644 {
1645   \bool_if:NF \l__enumext_store_active_bool
1646   {
1647     \msg_error:nnnn { enumext } { anskey-wrong-place } { anskey } { enumext }
1648   }
1649   \int_compare:nNnT { \l__enumext_keyans_level_int } = { 1 }
1650   {
1651     \msg_error:nnnn { enumext } { command-wrong-place } { anskey } { keyans }
1652   }
1653   \int_compare:nNnT { \l__enumext_keyans_pic_level_int } = { 1 }
1654   {
1655     \msg_error:nnnn { enumext } { command-wrong-place } { anskey } { keyanspic }
1656   }
1657   \group_begin:
1658     \bool_if:NF \l__enumext_store_ans_bool
1659     {
1660       \bool_if:NT \l__enumext_check_ans_bool
1661       {
1662         \int_gincr:N \g__enumext_count_item_ans_int
1663       }
1664       \__enumext_store_anskey_code:nnnn {#1} {#2} {#3} {#4}
1665     }
1666 }
```

```

1666     \group_end:
1667 }

```

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

```
\__enumext_store_anskey_code:nnnn
```

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

```

1668 \cs_new_protected:Npn \__enumext_store_anskey_code:nnnn #1 #2 #3 #4
1669 {
1670     \__enumext_store_addto_prop:n {#4}
1671     \bool_if:NT \l__enumext_store_ref_key_bool
1672     {
1673         \__enumext_store_internal_ref:
1674     }
1675     \__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`.

```

1676     \tl_clear:N \l__enumext_store_anskey_arg_tl
1677     \tl_if_novalue:nF {#3}
1678     {
1679         \keys_set:nn { enumext / anskey } {#3}
1680     }
1681     \bool_lazy_and:nnT
1682     { \l__enumext_store_columns_break_bool }
1683     { \bool_not_p:n { \l__enumext_starred_bool } }
1684     {
1685         \tl_put_left:Nn \l__enumext_store_anskey_arg_tl { \columnbreak }
1686     }
1687     \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).

```

1688     \tl_if_novalue:nF {#1}
1689     {
1690         \int_set:Nn \l__enumext_store_columns_join_int {#1}
1691         \bool_if:NT \l__enumext_starred_bool
1692         {
1693             \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
1694             {
1695                 ( \exp_not:V \l__enumext_store_columns_join_int )
1696             }
1697         }
1698     }

```

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`.

```

1699     \bool_if:nTF {#2}
1700     {
1701         \tl_put_right:Nn \l__enumext_store_anskey_arg_tl { * }
1702         \tl_if_empty:NF \l__enumext_store_item_symbol_tl
1703         {
1704             \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
1705             {
1706                 [ \exp_not:V \l__enumext_store_item_symbol_tl ]
1707             }
1708         }
1709         \dim_compare:nT
1710         {
1711             \l__enumext_store_item_symbol_sep_dim != \c_zero_dim
1712         }
1713         {
1714             \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
1715             {
1716                 [ \exp_not:V \l__enumext_store_item_symbol_sep_dim ]

```

```

1717         }
1718     }
1719     \tl_put_right:Nn \l__enumext_store_anskey_arg_tl {#4}
1720 }
1721 {
1722     \tl_put_right:Nn \l__enumext_store_anskey_arg_tl {#4}
1723 }

```

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`.

```

1724 \bool_lazy_and:nnT
1725 { \l__enumext_store_ref_key_bool }
1726 { \l__enumext_hyperref_bool }
1727 {
1728     \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
1729     {
1730         \hfill \exp_not:N \hyperlink { \exp_not:V \l__enumext_newlabel_arg_one_tl }
1731         { \exp_not:V \l__enumext_mark_ref_sym_tl }
1732     }
1733 }
1734 \__enumext_store_addto_seq:V \l__enumext_store_anskey_arg_tl
1735 }

```

(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`.

```

1736 \cs_new_protected:Nn \__enumext_store_internal_ref:
1737 {
1738     \cs_set_protected:Npn \__enumext_tmp:n ##1
1739     {
1740         \tl_set_eq:cc { \l__enumext_label_copy_##1_tl } { \l__enumext_label_##1_tl }
1741         \tl_reverse:c { \l__enumext_label_copy_##1_tl }
1742         \tl_remove_once:cn { \l__enumext_label_copy_##1_tl } { . }
1743         \tl_reverse:c { \l__enumext_label_copy_##1_tl }
1744     }
1745     \clist_map_inline:nn { i, ii, iii, iv, vii } { \__enumext_tmp:n {##1} }
1746     \cs_set:Npn \__enumext_tmp:n ##1
1747     { . \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.

```

1748 \bool_lazy_all:nT
1749 {
1750     { \g__enumext_starred_bool }
1751     { \int_compare_p:nNn { \l__enumext_level_int } = { \c_zero_int } }
1752 }
1753 {
1754     \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
1755     { \tl_use:N \l__enumext_label_copy_vii_tl }
1756 }
1757 \bool_lazy_all:nT
1758 {
1759     { \l__enumext_standar_bool }
1760     { \g__enumext_starred_bool }
1761     { \int_compare_p:nNn { \l__enumext_level_int } > { \c_zero_int } }
1762 }
1763 {
1764     \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
1765     {
1766         \tl_use:N \l__enumext_label_copy_vii_tl
1767         \int_step_function:nnN { 1 } { { \l__enumext_level_int } \__enumext_tmp:n
1768     }
1769 }

```

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.

```

1770 \bool_lazy_all:nT
1771 {

```

```

1772     { \l__enumext_standar_bool }
1773     { \int_compare_p:nNn { \l__enumext_level_int } > { \c_zero_int } }
1774     { \int_compare_p:nNn { \l__enumext_level_h_int } = { \c_zero_int } }
1775     { \bool_not_p:n { \l__enumext_starred_bool } }
1776   }
1777   {
1778     \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
1779     {
1780       \tl_use:N \l__enumext_label_copy_i_tl
1781       \int_step_function:nnN { 2 } { \l__enumext_level_int } \__enumext_tmp:n
1782     }
1783   }
1784   \cs_set:Npn \__enumext_tmp:n ##1
1785   { \tl_use:c { \l__enumext_label_copy_ \int_to_roman:n {##1} _tl } }
1786   \bool_lazy_all:nT
1787   {
1788     { \l__enumext_standar_bool }
1789     { \int_compare_p:nNn { \l__enumext_level_int } > { \c_zero_int } }
1790     { \bool_not_p:n { \g__enumext_starred_bool } }
1791     { \int_compare_p:nNn { \l__enumext_level_h_int } > { \c_zero_int } }
1792   }
1793   {
1794     \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
1795     {
1796       \int_step_function:nnN { 1 } { \l__enumext_level_int } \__enumext_tmp:n
1797       . \tl_use:N \l__enumext_label_copy_vii_tl
1798     }
1799   }

```

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

```

1800   \tl_put_right:Ne \l__enumext_newlabel_arg_one_tl
1801   {
1802     \l__enumext_store_name_tl \c_colon_str
1803     \int_eval:n { \prop_count:c { \g__enumext_ \l__enumext_store_name_tl _prop } }
1804   }

```

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.

```

1805   \tl_put_right:Ne \l__enumext_store_write_aux_file_tl
1806   {
1807     \__enumext_newlabel:nn
1808     { \exp_not:V \l__enumext_newlabel_arg_one_tl }
1809     { \l__enumext_newlabel_arg_two_tl }
1810   }
1811   \l__enumext_store_write_aux_file_tl
1812 }

```

(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.

```

1813 \cs_new_protected:Npn \__enumext_store_anskey_show_wrap:n #1
1814 {
1815   \par
1816   \bool_if:NT \l__enumext_starred_bool
1817   {
1818     \cs_set:Nn \__enumext_level: { vii }
1819   }
1820   \__enumext_print_keyans_box:cc
1821   { \l__enumext_labelwidth_ \__enumext_level: _dim }
1822   { \l__enumext_labelsep_ \__enumext_level: _dim }
1823   \__enumext_anskey_wrapper:n { #1 }
1824 }

```

(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.

```

1825 \cs_new_protected:Npn \__enumext_store_anskey_show_left:n #1
1826 {
1827   \bool_if:NT \l__enumext_show_answer_bool
1828   {
1829     \__enumext_store_anskey_show_wrap:n { #1 }
1830   }
1831   \bool_if:NT \l__enumext_show_position_bool
1832   {
1833     \tl_set:Nx \l__enumext_mark_answer_sym_tl
1834     {
1835       \group_begin:
1836       \exp_not:N \normalfont
1837       \exp_not:N \footnotesize [ \int_eval:n
1838       {
1839         \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
1840       }
1841       ]
1842       \group_end:
1843     }
1844     \__enumext_store_anskey_show_wrap:n { #1 }
1845   }
1846 }

```

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

11.25 Common functions for keyans and keyanspic

11.25.1 Storing content in prop list

`__enumext_keyans_addto_prop:n` The function `__enumext_keyans_addto_prop:n` will pass the contents of the `\l__enumext_label_v_tl` (current *label*) for the `keyans` environment and the `\l__enumext_label_vi_tl` (current *label*) for the `keyanspic` environment when using `\item*` and `\anspic*`, followed by the *contents* of the optional argument of both commands to the `\l__enumext_store_keyans_label_tl` variable, which will be passed to the *prop list* defined by the `save-ans` key using the `__enumext_store_addto_prop:V`.

```

1847 \cs_new_protected:Npn \__enumext_keyans_addto_prop:n #1
1848 {
1849   \tl_clear:N \l__enumext_store_keyans_label_tl
1850   \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
1851   {
1852     \tl_put_right:Nx \l__enumext_store_keyans_label_tl { \l__enumext_label_vi_tl }
1853   }
1854   {
1855     \tl_put_right:Nx \l__enumext_store_keyans_label_tl { \l__enumext_label_v_tl }
1856   }
1857   \tl_if_novalue:nF { #1 }
1858   {
1859     \tl_put_right:Nx \l__enumext_store_keyans_label_tl { \c_space_tl #1 }
1860   }
1861   \__enumext_store_addto_prop:V \l__enumext_store_keyans_label_tl
1862 }

```

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

11.25.2 The store-ref key for keyans and keyanspic

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

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

```

1863 \cs_new_protected:Nn \__enumext_keyans_internal_ref:
1864 {

```

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`.

```

1865   \cs_set_protected:Npn \__enumext_tmp:n ##1
1866   {
1867     \tl_set_eq:cc { \l__enumext_label_copy_##1_tl } { \l__enumext_label_##1_tl }
1868     \tl_reverse:c { \l__enumext_label_copy_##1_tl }
1869     \tl_remove_once:cn { \l__enumext_label_copy_##1_tl } { . }

```

```

1870         \tl_reverse:c { l__enumext_label_copy_##1_tl }
1871     }
1872     \clist_map_inline:nn { i, v, vi, vii, viii } { \__enumext_tmp:n {##1} }
1873     \int_compare:nNnTF { \__enumext_keyans_pic_level_int } = { 1 }
1874     {
1875         \tl_put_right:Ne \__enumext_newlabel_arg_two_tl
1876         { \__enumext_label_copy_i_tl . \__enumext_label_copy_vi_tl }
1877     }
1878     {
1879         \tl_put_right:Ne \__enumext_newlabel_arg_two_tl
1880         { \__enumext_label_copy_i_tl . \__enumext_label_copy_v_tl }
1881     }

```

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

```

1882     \tl_put_right:Ne \__enumext_newlabel_arg_one_tl
1883     {
1884         \__enumext_store_name_tl \c_colon_str
1885         \int_eval:n { \prop_count:c { g__enumext_ \__enumext_store_name_tl _prop } }
1886     }

```

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

```

1887     \tl_put_right:Ne \__enumext_store_write_aux_file_tl
1888     {
1889         \__enumext_newlabel:nn
1890         { \exp_not:V \__enumext_newlabel_arg_one_tl }
1891         { \__enumext_newlabel_arg_two_tl }
1892     }
1893     \bool_if:NT \__enumext_store_ref_key_bool
1894     {
1895         \__enumext_store_write_aux_file_tl
1896     }
1897 }

```

(End of definition for `__enumext_keyans_internal_ref:`)

11.25.3 Storing content in sequence

`__enumext_keyans_addto_seq:n`

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

```

1898 \cs_new_protected:Npn \__enumext_keyans_addto_seq:n #1
1899 {
1900     \tl_clear:N \__enumext_store_keyans_label_tl
1901     \int_compare:nNnTF { \__enumext_keyans_pic_level_int } = { 1 }
1902     {
1903         \tl_put_right:Ne \__enumext_store_keyans_label_tl { \item \__enumext_label_vi_tl }
1904     }
1905     {
1906         \tl_put_right:Ne \__enumext_store_keyans_label_tl { \item \__enumext_label_v_tl }
1907     }
1908     \tl_if_novalue:nF { #1 }
1909     {
1910         \tl_put_right:Ne \__enumext_store_keyans_label_tl { \c_space_tl #1 }
1911     }

```

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.

```

1912     \bool_lazy_and:nnT
1913     { \__enumext_store_ref_key_bool }
1914     { \__enumext_hyperref_bool }
1915     {
1916         \tl_put_right:Ne \__enumext_store_keyans_label_tl
1917         {
1918             \hfill \exp_not:N \hyperlink
1919             {
1920                 \exp_not:V \__enumext_newlabel_arg_one_tl
1921             }
1922             { \exp_not:V \__enumext_mark_ref_sym_tl }
1923         }

```



```

1924     }
1925     \__enumext_store_addto_seq:V \l__enumext_store_keyans_label_tl

```

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

```

1926     \tl_gset:NV \g__enumext_check_ans_item_tl \l__enumext_store_keyans_label_tl
1927     \bool_if:NT \l__enumext_check_ans_bool
1928     {
1929         \int_gincr:N \g__enumext_count_item_ans_int
1930     }
1931 }

```

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

11.25.4 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.

```

1932 \cs_new_protected:Npn \__enumext_keyans_check_ans:nn #1 #2
1933 {
1934     \tl_if_empty:NTF \g__enumext_check_ans_item_tl
1935     {
1936         \msg_warning:nnnn { enumext } { missing-starred }{ #1 }{ #2 }
1937     }
1938     { \tl_gclear:N \g__enumext_check_ans_item_tl }
1939 }

```

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

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

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

`__enumext_keyans_show_left:n`

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

```

1940 \cs_new_protected:Npn \__enumext_keyans_show_left:n #1
1941 {
1942     \bool_if:NT \l__enumext_show_answer_bool
1943     {
1944         \tl_put_left:Nn \l__enumext_label_v_tl
1945         {
1946             \__enumext_print_keyans_box:NN
1947             \l__enumext_labelwidth_i_dim
1948             \l__enumext_labelsep_i_dim
1949         }
1950         \tl_if_novalue:nF { #1 }
1951         { \tl_put_right:Nn \l__enumext_label_v_tl { \c_space_tl [ #1 ] } }
1952     }
1953     \bool_if:NT \l__enumext_show_position_bool
1954     {
1955         \tl_set:Nc \l__enumext_mark_answer_sym_tl
1956         {
1957             \group_begin:
1958             \exp_not:N \normalfont
1959             \exp_not:N \footnotesize [ \int_eval:n
1960             {
1961                 \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
1962                 + \l__enumext_keyans_level_int
1963             }
1964             ]
1965             \group_end:
1966         }
1967         \tl_put_left:Nn \l__enumext_label_v_tl
1968         {
1969             \__enumext_print_keyans_box:NN
1970             \l__enumext_labelwidth_i_dim
1971             \l__enumext_labelsep_i_dim
1972         }
1973         \tl_if_novalue:nF { #1 }

```

```

1974         { \tl_put_right:Nn \__enumext_label_v_tl { \c_space_tl [ #1 ] } }
1975     }
1976 }

```

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

11.26 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*`.

```

1977 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1978 {
1979     \keys_define:nn { enumext / #1 }
1980     {
1981         item-sym* .tl_set:c = { \__enumext_item_symbol_#2_tl },
1982         item-sym* .value_required:n = true,
1983         item-sym* .initial:n = { $\star$ },
1984         item-pos* .dim_set:c = { \__enumext_item_symbol_sep_#2_dim },
1985         item-pos* .value_required:n = true,
1986     }
1987 }
1988 \clist_map_inline:nn
1989 {
1990     {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {enumext*}{vii}
1991 }
1992 { \__enumext_tmp:nn #1 }

```

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

11.27 Redefining `\footnote` command

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

```

1993 \cs_new_protected:Nn \__enumext_footnotetext:nn
1994 {
1995     \footnotetext[#1]{#2}
1996 }
1997 \cs_new_protected:Nn \__enumext_renew_footnote:
1998 {
1999     \seq_gclear:N \g__enumext_footnote_arg_seq
2000     \seq_gclear:N \g__enumext_footnote_int_seq
2001     \RenewDocumentCommand \footnote { o +m }
2002     {
2003         \tl_if_novalue:nTF {##1}
2004         {
2005             \stepcounter{footnote}
2006             \int_gset_eq:Nc \g__enumext_footnote_int { c@footnote }
2007         }
2008         {
2009             \int_gset:Nn \g__enumext_footnote_int { ##1 }
2010         }
2011         \footnotemark [ \g__enumext_footnote_int ]
2012         \seq_gput_right:Nn \g__enumext_footnote_arg_seq { ##2 }
2013         \seq_gput_right:NV \g__enumext_footnote_int_seq \g__enumext_footnote_int
2014     }
2015 }
2016 \cs_new_protected:Nn \__enumext_print_footnote:
2017 {
2018     \seq_if_empty:NF \g__enumext_footnote_int_seq
2019     {
2020         \seq_map_pairwise_function:NNN
2021         \g__enumext_footnote_int_seq
2022         \g__enumext_footnote_arg_seq
2023         \__enumext_footnotetext:nn
2024     }
2025 }

```

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

11.28 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.

11.28.1 The \item command in enumext

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

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

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

The boolean variable `\l__enumext_wrap_label_X_bool` is used by the function `__enumext_make_label: (§11.29)`.

```

2026 \cs_new_protected:Npn \__enumext_default_item:n
2027 {
2028   \tl_if_novalue:nTF {#1}
2029   {
2030     \bool_if:NT \l__enumext_check_ans_bool
2031     {
2032       \int_gincr:N \g__enumext_count_item_all_int
2033       \int_gincr:c { g__enumext_count_level_ \__enumext_level: _int }
2034     }
2035     \bool_set_true:c { l__enumext_wrap_label_ \__enumext_level: _bool }
2036     \__enumext_item_std:w \tl_use:c { l__enumext_fake_item_indent_ \__enumext_level: _tl }
2037   }
2038   {
2039     \bool_set_eq:cc
2040     { l__enumext_wrap_label_ \__enumext_level: _bool }
2041     { l__enumext_wrap_label_opt_ \__enumext_level: _bool }
2042     \__enumext_item_std:w [#1] \tl_use:c { l__enumext_fake_item_indent_ \__enumext_level: _tl }
2043   }
2044 }

```

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

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

#1: `\l__enumext_item_symbol_X_tl`

#2: `\l__enumext_item_symbol_sep_X_dim`

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

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

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

The boolean variable `\l__enumext_wrap_label_X_bool` and the vars `\l__enumext_item_symbol_sep_X_dim`, `\g__enumext_item_symbol_tl` are used by the function `__enumext_make_label: (§11.29)`.

```

2045 \cs_new_protected:Npn \__enumext_starred_item:nn #1 #2
2046 {
2047   \tl_if_novalue:nF {#1}
2048   {
2049     \tl_set:cn { l__enumext_item_symbol_ \__enumext_level: _tl } {#1}
2050   }
2051   \tl_gset_eq:Nc \g__enumext_item_symbol_tl { l__enumext_item_symbol_ \__enumext_level: _tl }
2052   \tl_if_novalue:nTF {#2}
2053   {
2054     \dim_set_eq:cc
2055     { l__enumext_item_symbol_sep_ \__enumext_level: _dim }
2056     { l__enumext_labelsep_ \__enumext_level: _dim }
2057   }

```

```

2058     {
2059         \dim_set:cn { l__enumext_item_symbol_sep_ \__enumext_level: _dim } {#2}
2060     }
2061     \bool_if:NT \l__enumext_check_ans_bool
2062     {
2063         \int_gincr:N \g__enumext_count_item_all_int
2064         \int_gincr:c { g__enumext_count_level_ \__enumext_level: _int }
2065     }
2066     \bool_set_true:c { l__enumext_wrap_label_ \__enumext_level: _bool }
2067     \__enumext_item_std:w \tl_use:c { l__enumext_fake_item_indent_ \__enumext_level: _tl }
2068 }

```

(End of definition for __enumext_starred_item:nn.)

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

This function is passed to `__enumext_list_arg_two_X:` which is used in the definition of the `enumext` environment (§11.31).

```

2069 \cs_new_protected:Nn \__enumext_redefine_item:
2070 {
2071     \RenewDocumentCommand \item { s o o }
2072     {
2073         \bool_if:nTF {##1}
2074         {
2075             \__enumext_starred_item:nn {##2} {##3}
2076         }
2077         { \__enumext_default_item:n {##2} }
2078     }
2079 }

```

(End of definition for __enumext_redefine_item:.)

11.28.2 The `\item` command in keyans

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

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

```

2080 \cs_new_protected:Npn \__enumext_keyans_default_item:n #1
2081 {
2082     \tl_if_novalue:nTF { #1 }
2083     {
2084         \bool_set_true:N \l__enumext_wrap_label_v_bool
2085         \__enumext_item_std:w \tl_use:N \l__enumext_fake_item_indent_v_tl
2086     }
2087     {
2088         \bool_set_eq:NN \l__enumext_wrap_label_v_bool \l__enumext_wrap_label_opt_v_bool
2089         \__enumext_item_std:w [#1] \tl_use:N \l__enumext_fake_item_indent_v_tl
2090     }
2091 }

```

(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*.

```

2092 \cs_new_protected:Npn \__enumext_keyans_starred_item:n #1
2093 {
2094     \tl_set_eq:NN \l__enumext_keyans_tmpa_tl \l__enumext_label_v_tl
2095     \__enumext_keyans_show_left:n { #1 }
2096     \bool_set_true:N \l__enumext_wrap_label_v_bool
2097     \__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*.

```

2098     \tl_set_eq:NN \l__enumext_label_v_tl \l__enumext_keyans_tmpa_tl
2099     \__enumext_keyans_addto_prop:n { #1 }
2100     \__enumext_keyans_internal_ref:
2101     \__enumext_keyans_addto_seq:n { #1 }
2102 }

```

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

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

This function is passed to `__enumext_list_arg_two_v:` which is used in the definition of the `keyans` environment (§11.31).

```

2103 \cs_new_protected:Nn \__enumext_keyans_redefine_item:
2104 {
2105   \RenewDocumentCommand \item { s o }
2106   {
2107     \bool_if:nTF {##1}
2108     {
2109       \peek_remove_spaces:n
2110       {
2111         \__enumext_keyans_starred_item:n {##2}
2112       }
2113     }
2114     {
2115       \__enumext_keyans_default_item:n {##2}
2116     }
2117   }
2118 }

```

(End of definition for `\item*` and `__enumext_keyans_redefine_item:`. This function is documented on page 10.)

11.29 Redefining `\makeLabel` command

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

11.29.1 Redefining `\makeLabel` for `enumext`

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

```

2119 \cs_new_protected:Nn \__enumext_item_starred:
2120 {
2121   \tl_if_empty:cF { l__enumext_item_symbol_ \__enumext_level: _tl }
2122   {
2123     \mode_leave_vertical:
2124     \skip_horizontal:n { -\dim_use:c { l__enumext_item_symbol_sep_ \__enumext_level: _dim } }
2125     \makebox[ 0pt ][ r ]{ \tl_use:N \g__enumext_item_symbol_tl }
2126     \skip_horizontal:n { \dim_use:c { l__enumext_item_symbol_sep_ \__enumext_level: _dim } }
2127   }
2128 }

```

(End of definition for `__enumext_item_starred:`.)

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

This function is passed to `__enumext_list_arg_two_X:` which is used in the definition of the `enumext` environment (§11.31).

```

2129 \cs_new_protected:Nn \__enumext_make_label:
2130 {
2131   \RenewDocumentCommand \makeLabel { m }
2132   {
2133     \tl_use:c { l__enumext_label_fill_left_ \__enumext_level: _tl }
2134     \tl_use:c { l__enumext_label_font_style_ \__enumext_level: _tl }
2135     \bool_if:cTF { l__enumext_wrap_label_ \__enumext_level: _bool }
2136     {
2137       \__enumext_item_starred:
2138       \use:c { __enumext_wrapper_label_ \__enumext_level: :n } { ##1 }
2139     }
2140     { ##1 }
2141     \tl_use:c { l__enumext_label_fill_right_ \__enumext_level: _tl }
2142     \tl_gclear:N \g__enumext_item_symbol_tl
2143   }
2144 }

```

(End of definition for `__enumext_make_label:`.)

11.29.2 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 (§11.31).

```

2145 \cs_new_protected:Nn \__enumext_keyans_make_label:
2146 {
2147   \RenewDocumentCommand \makeLabel { m }
2148   {
2149     \tl_use:N \l__enumext_label_fill_left_v_tl
2150     \tl_use:N \l__enumext_label_font_style_v_tl
2151     \bool_if:NTF \l__enumext_wrap_label_v_bool
2152     {
2153       \__enumext_wrapper_label_v:n { ##1 }
2154     }
2155     { ##1 }
2156     \tl_use:N \l__enumext_label_fill_right_v_tl
2157   }
2158 }

```

(End of definition for `__enumext_keyans_make_label:`)

11.30 Calculation of `\leftmargin` and `\itemindent`

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

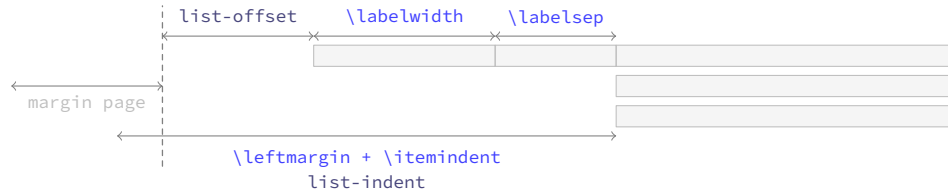


Figure 9: Representation of standard horizontal lengths in `list` environment.

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

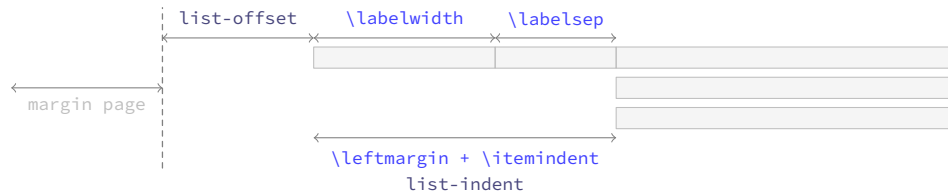


Figure 10: Representation of horizontal lengths concept in list in `enumext`.

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

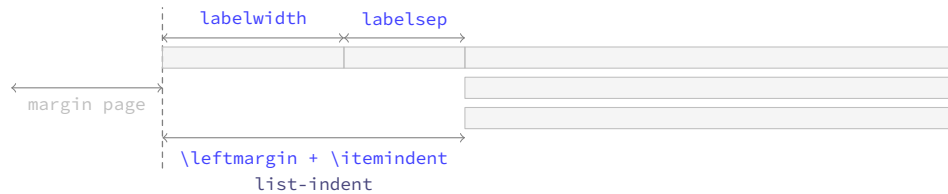


Figure 11: Default horizontal lengths in `enumext`.

```

\__enumext_calc_hspace:NNNNNN
\__enumext_calc_hspace:cccccc

```

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

```

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

```

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

This function is passed to `__enumext_list_arg_two_X:` which is used in the definition of the `enumext` and `keyans` environments (§11.31).

```

2159 \cs_new_protected:Npn \__enumext_calc_hspace:NNNNNN #1#2#3#4#5#6#7
2160 {
2161   \dim_compare:nNnT { #1 } < { \c_zero_dim }

```

```

2162     {
2163       \msg_warning:nnnV { enumext } { width-non-positive }{ labelwidth }{ #1 }
2164       \dim_set:Nn #1 { \dim_abs:n { #1 } }
2165     }
2166     \dim_compare:nNnT { #2 } < { \c_zero_dim }
2167     {
2168       \msg_warning:nnnV { enumext } { width-negative }{ labelsep }{ #2 }
2169       \dim_set:Nn #2 { \dim_abs:n { #2 } }
2170     }

```

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

```

2171     \bool_if:nF #7 { \dim_set:Nn #4 { #1 + #2 } }

```

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

```

2172     \dim_compare:nNnTF { #4 } < { \c_zero_dim }
2173     {
2174       \dim_set:Nn #6 { #1 + #2 - #4 }
2175       \dim_set:Nn #5 { #1 + #2 + #3 - #6 }
2176     }
2177     {
2178       \dim_compare:nNnT { #4 } = { #1 + #2 }
2179       { \dim_set:Nn #6 { \c_zero_dim } }
2180       \dim_compare:nNnT { #4 } < { #1 + #2 }
2181       { \dim_set:Nn #6 { #1 + #2 - #4 } }
2182       \dim_compare:nNnT { #4 } > { #1 + #2 }
2183       {
2184         \dim_set:Nn #6 { -#1 - #2 + #4 }
2185         \dim_set:Nn #6 { #6*-1 }
2186       }
2187       \dim_set:Nn #5 { #1 + #2 + #3 - #6 }
2188     }
2189   }
2190   \cs_generate_variant:Nn \__enumext_calc_hspace:NNNNNNN { ccccccc }

```

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

11.31 Setting second argument of the lists

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

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

We will “not set” `\leftmargini`, `\leftmarginii`, `\leftmarginiii` or `\leftmarginiv`, in this case, we will directly set the parameters for vertical and horizontal list spacing per level.

```

2191   \cs_set_protected:Npn \__enumext_tmp:n #1
2192   {
2193     \cs_new_protected:cpn { __enumext_list_arg_two_#1: }
2194     {
2195       \__enumext_calc_hspace:cccccc
2196       { \__enumext_labelwidth_#1_dim } { \__enumext_labelsep_#1_dim }
2197       { \__enumext_listoffset_#1_dim } { \__enumext_leftmargin_tmp_#1_dim }
2198       { \__enumext_leftmargin_#1_dim } { \__enumext_itemindent_#1_dim }
2199       { \__enumext_leftmargin_tmp_#1_bool }
2200       \clist_map_inline:nn
2201       { labelsep, labelwidth, itemindent, leftmargin, rightmargin, listparindent }
2202       { \dim_set_eq:cc {####1} { \__enumext_####1_#1_dim } }
2203       \clist_map_inline:nn { topsep, parsep, partopsep, itemsep }
2204       { \skip_set_eq:cc {####1} { \__enumext_####1_#1_skip } }
2205       \usecounter { enumX#1 }
2206       \bool_lazy_and:nnTF
2207       { \str_if_eq_p:nn {#1} { i } }
2208       { \bool_if_p:N \__enumext_resume_bool }
2209       { \setcounter { enumXi } { \int_eval:n { \g__enumext_resume_int } } }
2210       {
2211         \setcounter { enumX#1 }
2212         { \int_eval:n { \int_use:c { \__enumext_start_#1_int } - 1 } }
2213       }
2214       \str_if_eq:nnTF {#1} { v }

```



```

2215     {
2216         \__enumext_keyans_redefine_item:
2217         \__enumext_keyans_make_label:
2218         \__enumext_keyans_fake_item:
2219         \bool_if:cT { \__enumext_show_length_#1_bool }
2220         {
2221             \msg_term:nnnn { enumext } { list-lengths-not-nested } { v } { keyans }
2222         }
2223     }
2224     {
2225         \__enumext_redefine_item:
2226         \__enumext_make_label:
2227         \__enumext_use_key_ref:
2228         \__enumext_fake_item:
2229         \bool_if:cT { \__enumext_show_length_#1_bool }
2230         {
2231             \msg_term:nnne { enumext } { list-lengths } {#1} { \int_use:N \__enumext_level\i
2232         }
2233     }
2234 }
2235 }
2236 \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.

```

2237 \cs_set_protected:Npn \__enumext_tmp:n #1
2238 {
2239     \cs_new_protected:cpn { __enumext_list_arg_two_#1: }
2240     {
2241         \__enumext_calc_hspace:ccccc
2242         { \__enumext_labelwidth_#1_dim } { \__enumext_labelsep_#1_dim }
2243         { \__enumext_listoffset_#1_dim } { \__enumext_leftmargin_tmp_#1_dim }
2244         { \__enumext_leftmargin_#1_dim } { \__enumext_itemindent_#1_dim }
2245         { \__enumext_leftmargin_tmp_#1_bool }
2246         \clist_map_inline:nn
2247         { labelsep, labelwidth, itemindent, leftmargin, rightmargin, listparindent }
2248         { \dim_set_eq:cc {####1} { \__enumext_####1_#1_dim } }
2249         \clist_map_inline:nn { topsep, parsep, partopsep, itemsep }
2250         { \skip_set_eq:cc {####1} { \__enumext_####1_#1_skip } }
2251         \skip_set_eq:Nc \parsep { \__enumext_itemsep_#1_skip }
2252         \skip_zero:N \partopsep
2253         \usecounter { enumX#1 }
2254         \bool_lazy_and:nnTF
2255         { \str_if_eq_p:nn {#1} { vii } } { \bool_if_p:N \__enumext_resume_vii_bool }
2256         { \setcounter { enumXvii } { \int_eval:n { \g__enumext_resume_vii_int } } }
2257         {
2258             \setcounter { enumX#1 }
2259             { \int_eval:n { \int_use:c { \__enumext_start_#1_int } - 1 } }
2260         }
2261         \__enumext_use_key_ref_h:
2262         \str_if_eq:nnTF {#1} { vii }
2263         {
2264             \__enumext_fake_item_vii:
2265             \bool_if:cT { \__enumext_show_length_vii_bool }
2266             { \msg_term:nnnn { enumext } { list-lengths-not-nested } { vii } { enumext* } }
2267         }
2268         {
2269             \__enumext_fake_item_viii:
2270             \bool_if:cT { \__enumext_show_length_#1_bool }
2271             { \msg_term:nnnn { enumext } { list-lengths-not-nested } { #1 } { keyans* } }
2272         }
2273     }
2274 }
2275 \clist_map_inline:nn { vii, viii } { \__enumext_tmp:n {#1} }

```

(End of definition for __enumext_list_arg_two_vii: and __enumext_list_arg_two_viii:.)

11.32 The environment enumext

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

```

2276 \NewDocumentEnvironment{enumext}{ 0{ } }
2277 {
2278   \__enumext_safe_exec:
2279   \__enumext_parse_keys:n {#1}
2280   \__enumext_before_list:
2281   \__enumext_start_store_level:
2282   \__enumext_start_list:nn
2283   { \tl_use:c { l__enumext_label_ \__enumext_level: _tl } }
2284   {
2285     \use:c { __enumext_list_arg_two_ \__enumext_level: : }
2286     \__enumext_before_keys_exec:
2287   }
2288   \__enumext_after_args_exec:
2289 }
2290 {
2291   \__enumext_stop_list:
2292   \__enumext_stop_store_level:
2293   \__enumext_after_list:
2294 }

```

(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.

```

2295 \cs_new_protected:Nn \__enumext_safe_exec:
2296 {
2297   \int_incr:N \l__enumext_level_int
2298   \int_compare:nNnT { \l__enumext_level_int } > { 4 }
2299   { \msg_fatal:nn { enumext } { list-too-deep } }
2300   \bool_set_true:N \l__enumext_standar_bool
2301   \bool_lazy_all:nT
2302   {
2303     { \bool_not_p:n { \l__enumext_starred_bool } }
2304     { \int_compare_p:nNn { \l__enumext_level_h_int } = { \c_zero_int } }
2305   }
2306   {
2307     \bool_gset_true:N \g__enumext_standar_bool
2308   }
2309 }

```

(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.

```

2310 \cs_new_protected:Npn \__enumext_parse_keys:n #1
2311 {
2312   \exp_args:Ne \keys_set:nn
2313   { enumext / level-\int_use:N \l__enumext_level_int } {#1}
2314   \bool_if:NT \l__enumext_store_active_bool
2315   {
2316     \__enumext_parse_store_keys:n {#1}
2317   }
2318 }

```

(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:.**

```

2319 \cs_new_protected:Npn \__enumext_parse_store_keys:n #1
2320 {
2321   \bool_if:cF { l__enumext_store_columns_ \__enumext_level: _bool }
2322   {
2323     \regex_match:nnT { \b columns\b } {#1}
2324     {
2325       \int_set_eq:cc

```

```

2326         { l__enumext_store_columns_ \__enumext_level: _int }
2327         { l__enumext_columns_ \__enumext_level: _int }
2328         \tl_put_right:ce { l__enumext_store_opt_ \__enumext_level: _tl }
2329         {
2330             columns = \exp_not:v { l__enumext_store_columns_ \__enumext_level: _int },
2331         }
2332     }
2333 }
2334 \bool_if:cF { l__enumext_store_columns_sep_ \__enumext_level: _bool }
2335 {
2336     \regex_match:nnT { \b columns-sep \b} {#1}
2337     {
2338         \dim_set_eq:cc
2339         { l__enumext_store_columns_sep_ \__enumext_level: _dim }
2340         { l__enumext_columns_sep_ \__enumext_level: _dim }
2341         \tl_put_right:ce { l__enumext_store_opt_ \__enumext_level: _tl }
2342         {
2343             columns-sep = \exp_not:v { l__enumext_store_columns_sep_ \__enumext_level: _dim }
2344         }
2345     }
2346 }
2347 }

```

(End of definition for __enumext_parse_store_keys:n.)

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

```

2348 \cs_new_protected:Nn \__enumext_start_store_level:
2349 {
2350     \bool_lazy_and:nnT { \__enumext_store_active_bool } { \bool_not_p:n { \__enumext_keyans_env: } }
2351     {
2352         \int_compare:nNnT { \__enumext_level_h_int } = { 1 }
2353         {
2354             \bool_set_true:c { l__enumext_store_upper_level_ \__enumext_level: _bool }
2355             \__enumext_store_level_open:
2356         }
2357         \int_compare:nNnT { \__enumext_level_int } > { 1 }
2358         {
2359             \bool_set_true:c { l__enumext_store_upper_level_ \__enumext_level: _bool }
2360             \__enumext_store_level_open:
2361         }
2362     }
2363 }
2364 \cs_new_protected:Nn \__enumext_stop_store_level:
2365 {
2366     \bool_if:cT { l__enumext_store_upper_level_ \__enumext_level: _bool }
2367     {
2368         \__enumext_store_level_close:
2369     }
2370 }

```

(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.

```

2371 \cs_new_protected:Nn \__enumext_before_list:
2372 {
2373     \__enumext_vspace_above:
2374     \__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.

```

2375     \__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.

```

2376 \dim_compare:nNtT
2377 { \dim_use:c { l__enumext_minipage_right_ \__enumext_level: _dim } } > { \c_zero_dim }
2378 {
2379   \dim_set:cn { l__enumext_minipage_left_ \__enumext_level: _dim }
2380   {
2381     \linewidth
2382     - \dim_use:c { l__enumext_minipage_right_ \__enumext_level: _dim }
2383     - \dim_use:c { l__enumext_minipage_hsep_ \__enumext_level: _dim }
2384   }

```

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

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

```

2385   \bool_set_true:c { l__enumext_minipage_active_ \__enumext_level: _bool }
2386   \int_gincr:N \g__enumext_minipage_stat_int
2387   \__enumext_mini_addvspace:
2388   \nointerlineskip\noindent
2389   \begin{\__enumext_mini_env*}
2390   { \dim_use:c { l__enumext_minipage_left_ \__enumext_level: _dim } }
2391   }
2392   \__enumext_multicols_start:
2393   }

```

(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.

```

2394 \cs_new_protected:Nn \__enumext_multicols_start:
2395 {
2396   \int_compare:nNtT
2397   { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } > { 1 }
2398   {
2399     \dim_compare:nNtT
2400     { \dim_use:c { l__enumext_columns_sep_ \__enumext_level: _dim } } = { \c_zero_dim }
2401     {
2402       \dim_set:cn { l__enumext_columns_sep_ \__enumext_level: _dim }
2403       {
2404         ( \dim_use:c { l__enumext_labelwidth_ \__enumext_level: _dim }
2405         + \dim_use:c { l__enumext_labelsep_ \__enumext_level: _dim }
2406         ) / \int_use:c { l__enumext_columns_ \__enumext_level: _int }
2407         - \dim_use:c { l__enumext_listoffset_ \__enumext_level: _dim }
2408       }
2409     }
2410     \dim_set_eq:Nc \columnsep { l__enumext_columns_sep_ \__enumext_level: _dim }
2411     \skip_zero:N \multicolsep
2412     \int_compare:nNtT { \l__enumext_level_int } > { 1 }
2413     {
2414       \dim_zero:N \columnseprule
2415     }

```

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.

```

2416   \bool_if:cF { l__enumext_minipage_active_ \__enumext_level: _bool }
2417   {
2418     \__enumext_multi_addvspace:
2419   }
2420   \raggedcolumns
2421   \begin{multicols}{ \int_use:c { l__enumext_columns_ \__enumext_level: _int } }
2422   }
2423   }

```

(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.

```

2424 \cs_new_protected:Nn \__enumext_multicols_stop:
2425 {
2426   \int_compare:nNt
2427     { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } > { 1 }
2428   {
2429     \end{multicols}
2430     \bool_if:cF { l__enumext_minipage_active_ \__enumext_level: _bool }
2431     {
2432       \par\addvspace{ \skip_use:c { l__enumext_multicols_below_ \__enumext_level: _skip } }
2433     }
2434   }

```

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.

```

2435   \bool_lazy_and:nnT { \l__enumext_check_ans_bool }{ \bool_not_p:n { \g__enumext_starred_bool }
2436   {
2437     \bool_gset_true:N \g__enumext_check_ans_show_bool
2438     \tl_gset:NV \g__enumext_store_name_tl \l__enumext_store_name_tl
2439   }
2440 }

```

(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.

```

2441 \cs_new_protected:Nn \__enumext_after_list:
2442 {
2443   \bool_if:cTF { l__enumext_minipage_active_ \__enumext_level: _bool }
2444   {
2445     \int_compare:nNt { \g__enumext_minipage_stat_int } = { 1 }
2446     {
2447       \msg_warning:nn { enumext } { missing-miniright }
2448       \miniright
2449     }
2450     \int_gzero:N \g__enumext_minipage_stat_int
2451     \end{__enumext_mini_env*}
2452     \par\addvspace { \l__enumext_minipage_after_skip }
2453   }
2454   { \__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.

```

2455   \__enumext_after_stop_list:
2456   \__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.

```

2457   \bool_set_false:N \l__enumext_standar_bool
2458   \int_gset_eq:NN \g__enumext_resume_int \value{enumXi}
2459   \int_if_exist:cT { g__enumext_resume_ \l__enumext_store_name_tl _int }
2460   {
2461     \int_gset_eq:cN
2462     { g__enumext_resume_ \l__enumext_store_name_tl _int }
2463     { \value{enumXi} }
2464   }
2465 }

```

(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`.

```

2466 \__enumext_after_env:nn {enumext}
2467 {

```

```

2468 \bool_if:NT \g__enumext_check_ans_show_bool
2469 {
2470     \int_compare:nNnT { \l__enumext_level_int } = { 0 }
2471     {
2472         \__enumext_check_ans_active:
2473     }
2474 }
2475 \bool_gset_false:N \g__enumext_check_ans_show_bool
2476 \tl_gclear:N \g__enumext_store_name_tl
2477 }

```

11.33 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.

```

2478 \NewDocumentEnvironment{keyans}{0}{}
2479 {
2480     \__enumext_keyans_safe_exec:
2481     \__enumext_keyans_parse_keys:n {#1}
2482     \__enumext_before_list_v:
2483     \__enumext_start_list:nn
2484     { \tl_use:N \l__enumext_label_v_tl }
2485     {
2486         \__enumext_list_arg_two_v:
2487         \__enumext_before_keys_exec_v:
2488     }
2489     \__enumext_after_args_exec_v:
2490 }
2491 {
2492     \__enumext_keyans_check_ans:nn { item }{ keyans }
2493     \__enumext_stop_list:
2494     \__enumext_after_list_v:
2495 }

```

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

`__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.

```

2496 \cs_new_protected:Nn \__enumext_keyans_safe_exec:
2497 {
2498     \bool_if:NF \l__enumext_store_active_bool
2499     {
2500         \msg_error:nnnn { enumext } { wrong-place }{ keyans }{ save-ans }
2501     }
2502     \int_incr:N \l__enumext_keyans_level_int
2503     \bool_set_true:N \l__enumext_keyans_env_bool
2504     % set false for interfering with enumext nested in keyans (yes, its posible and crayze)
2505     \bool_set_false:N \l__enumext_store_active_bool
2506     \int_compare:nNnT { \l__enumext_keyans_level_int } > { 1 }
2507     {
2508         \msg_error:nn { enumext } { keyans-nested }
2509     }
2510     \int_compare:nNnT { \l__enumext_level_int } > { 1 }
2511     {
2512         \msg_error:nn { enumext } { keyans-wrong-level }
2513     }
2514 }

```

(End of definition for `__enumext_keyans_safe_exec:.`)

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

```

2515 \cs_new_protected:Npn \__enumext_keyans_parse_keys:n #1
2516 {
2517     \keys_set:nn { enumext / keyans } {#1}
2518 }

```

(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.

```

2519 \cs_new_protected:Nn \__enumext_before_list_v:
2520 {
2521   \__enumext_vspace_above_v:
2522   \__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.

```

2523   \dim_compare:nNnT { \l__enumext_minipage_right_v_dim } > { \c_zero_dim }
2524   {
2525     \dim_set:Nn \l__enumext_minipage_left_v_dim
2526     {
2527       \linewidth - \l__enumext_minipage_right_v_dim - \l__enumext_minipage_hsep_v_dim
2528     }

```

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*.

```

2529   \bool_set_true:N \l__enumext_minipage_active_v_bool
2530   \int_gincr:N \g__enumext_minipage_stat_int
2531   \__enumext_keyans_mini_addvspace:
2532   \nointerlineskip\noindent
2533   \begin{\__enumext_mini_env*}{ \l__enumext_minipage_left_v_dim }
2534   }

```

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

```

2535   \__enumext_keyans_multicols_start:
2536   }

```

(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.

```

2537 \cs_new_protected:Nn \__enumext_keyans_multicols_start:
2538 {
2539   \int_compare:nNnT { \l__enumext_columns_v_int } > { 1 }
2540   {

```

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

```

2541     \dim_compare:nNnT { \l__enumext_columns_sep_v_dim } = { \c_zero_dim }
2542     {
2543       \dim_set:Nn \l__enumext_columns_sep_v_dim
2544       {
2545         (
2546           \l__enumext_labelwidth_v_dim + \l__enumext_labelsep_v_dim
2547         ) / \l__enumext_columns_v_int
2548         - \l__enumext_listoffset_v_dim
2549       }
2550     }
2551     \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).

```

2552     \skip_zero:N \multicolsep
2553     \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.

```

2554     \bool_if:NF \l__enumext_minipage_active_v_bool
2555     {
2556       \__enumext_keyans_multi_addvspace:
2557     }
2558     \raggedcolumns

```



```

2559     \begin{multicols}{\l__enumext_columns_v_int }
2560   }
2561 }

```

(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.

```

2562 \cs_new_protected:Nn \__enumext_keyans_multicols_stop:
2563 {
2564   \int_compare:nNtT { \l__enumext_columns_v_int } > { 1 }
2565   {
2566     \end{multicols}
2567     \bool_if:NF \l__enumext_minipage_active_v_bool
2568     {
2569       \par\addvspace{ \l__enumext_multicols_below_v_skip }
2570     }
2571   }
2572 }

```

(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.

```

2573 \cs_new_protected:Nn \__enumext_after_list_v:
2574 {
2575   \bool_if:NTF \l__enumext_minipage_active_v_bool
2576   {
2577     \int_compare:nNtT { \g__enumext_minipage_stat_int } = { 1 }
2578     {
2579       \msg_warning:nn { enumext } { missing-miniright }
2580       \miniright
2581     }
2582     \int_gzero:N \g__enumext_minipage_stat_int
2583     \end{\__enumext_mini-env*}
2584     \par\addvspace{ \l__enumext_minipage_after_skip }
2585   }
2586   { \__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.

```

2587   \bool_set_false:N \l__enumext_keyans_env_bool
2588   \__enumext_after_stop_list_v:
2589   \__enumext_vspace_below_v:
2590 }

```

(End of definition for `__enumext_after_list_v:`)

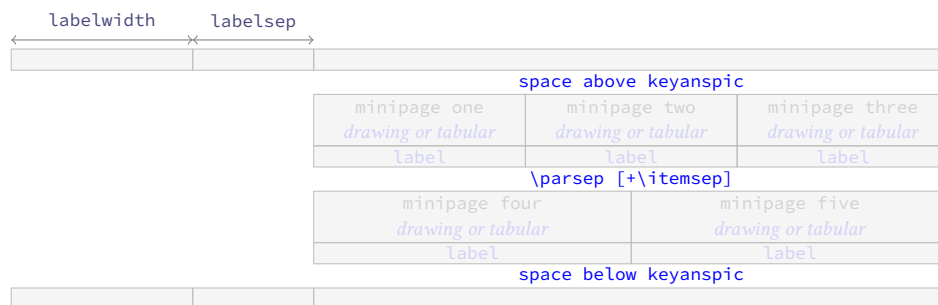
11.34 The environment `keyanspic` and `\anspic`

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

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

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

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

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

11.34.1 The command `\anspic`

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

```
2591 \NewDocumentCommand \anspic { s o +m }
2592 {
```

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

```
2593   \bool_if:NF \l__enumext_store_active_bool
2594   {
2595     \msg_error:nnnn { enumext } { wrong-place } { keyanspic } { save-ans }
2596   }
2597   \int_compare:nNnT { \l__enumext_level_int } > { 1 }
2598   {
2599     \msg_error:nn { enumext } { keyanspic-wrong-level }
2600   }
2601   \int_compare:nNnT { \l__enumext_keyans_level_int } = { 1 }
2602   {
2603     \msg_error:nnnn { enumext } { command-wrong-place } { anspic } { keyans }
2604   }
```

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

```
2605   \seq_put_right:Nn \l__enumext_keyans_pic_body_seq
2606   {
2607     \__enumext_keyans_anspic_code:nnn { #1 } { #2 } { #3 }
2608   }
2609 }
```

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

`__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.

```
2610 \cs_new_protected:Nn \__enumext_keyans_anspic_code:nnn
2611 {
2612   \stepcounter { enumXvi }
2613   #3 \\\
2614   \bool_if:nT { #1 }
2615   {
2616     \__enumext_keyans_addto_prop:n { #2 }
2617     \__enumext_keyans_internal_ref:
2618     \__enumext_keyans_addto_seq:n { #2 }
2619     \bool_lazy_or:nnT
2620     { \l__enumext_show_answer_bool }
2621     { \l__enumext_show_position_bool }
2622     {
2623       \tl_set_eq:NN \l__enumext_label_v_tl \l__enumext_label_vi_tl
2624       \__enumext_keyans_show_left:n { #2 }
2625       \tl_set_eq:NN \l__enumext_label_vi_tl \l__enumext_label_v_tl
2626     }
2627   }
2628   \tl_use:N \l__enumext_label_font_style_v_tl
2629   \__enumext_wrapper_label_v:n { \__enumext_label_vi_tl }
2630 }
```

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

11.34.2 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.

```
2631 \NewDocumentEnvironment{keyanspic}{ o }
2632 {
2633   \__enumext_keyans_pic_safe_exec:
2634   \__enumext_start_list:nn
2635   { }
2636   {
2637     \__enumext_keyans_pic_arg_two:
2638   }

```

We apply the “adjusted” vertical spacing above the environment

```
2639 \vspace { \__enumext_keyans_pic_above_skip }
2640 }
```

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

```
2641 {
2642   \tl_if_novalue:nTF { #1 }
2643   {
2644     \__enumext_keyans_pic_do:e { \seq_count:N \l__enumext_keyans_pic_body_seq }
2645   }
2646   { \__enumext_keyans_pic_do:n { #1 } }
2647   \__enumext_stop_list:
2648   \__enumext_keyans_check_ans:nn { anspic } { keyanspic }
2649   \setcounter { enumXvi } { 0 }
2650   \vspace { \__enumext_topsep_v_skip }
2651   %\bool_set_false:N \l__enumext_store_active_bool
2652 }
```

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

`__enumext_keyans_pic_safe_exec:`

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

```
2653 \cs_new_protected:Nn \__enumext_keyans_pic_safe_exec:
2654 {
2655   \int_incr:N \l__enumext_keyans_pic_level_int
2656   \int_compare:nNtT { \l__enumext_keyans_pic_level_int } > { 1 }
2657   {
2658     \msg_error:nn { enumext } { keyanspic-nested }
2659   }
2660 }
```

(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`.

```
2661 \cs_new_protected:Npn \__enumext_keyans_pic_skip_abs:N #1
2662 {
2663   \dim_compare:nNtT { #1 } < { 0pt }
2664   { \skip_set:Nn #1 { -#1 } }
2665 }
```

(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.

```
2666 \cs_new_protected:Nn \__enumext_keyans_pic_arg_two:
2667 {
```

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.

```
2668   \bool_set_false:N \l__enumext_leftmargin_tmp_v_bool
2669   \__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.

```

2670 \skip_add:Nn \parsep { \itemsep }
2671 \dim_add:Nn \leftmargin { -\labelwidth - \labelsep }
2672 \dim_zero:N \labelwidth
2673 \dim_zero:N \listparindent
2674 \dim_zero:N \labelsep
2675 \skip_zero:N \partopsep
2676 \skip_zero:N \itemsep

```

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

```

2677 \__enumxt_keyans_pic_skip_abs:N \parsep
2678 \skip_set:Nn \l__enumxt_keyans_pic_above_skip
2679 {
2680   \box_dp:N \strutbox
2681   + \l__enumxt_topsep_v_skip
2682   - \parsep
2683 }
2684 \__enumxt_item_std:w \scan_stop:
2685 }

```

(End of definition for `__enumxt_keyans_pic_arg_two:.`)

```

\__enumxt_keyans_pic_do:n
\__enumxt_keyans_pic_do:e

```

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

```

2686 \cs_new_protected:Nn \__enumxt_keyans_pic_do:n
2687 {
2688   \clist_map_function:nN { #1 } \__enumxt_keyans_pic_row:n
2689 }
2690 \cs_generate_variant:Nn \__enumxt_keyans_pic_do:n { e }

```

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

```
\__enumxt_keyans_pic_row:n
```

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

```

2691 \cs_new_protected:Nn \__enumxt_keyans_pic_row:n
2692 {
2693   \dim_set:Nn \l__enumxt_keyans_pic_width_dim { \linewidth / #1 }
2694   \int_set:Nn \l__enumxt_keyans_pic_above_int { \l__enumxt_keyans_pic_below_int }
2695   \int_set:Nn \l__enumxt_keyans_pic_below_int { \l__enumxt_keyans_pic_above_int + #1 }
2696   \int_step_inline:nnn
2697     { \l__enumxt_keyans_pic_above_int + 1 }
2698     { \l__enumxt_keyans_pic_below_int }
2699     {
2700       \__enumxt_minipage:w [ b ]{ \l__enumxt_keyans_pic_width_dim }
2701       \centering
2702       \seq_item:Nn \l__enumxt_keyans_pic_body_seq { ##1 }
2703       \__enumxt_endminipage:
2704     }
2705   \par
2706 }

```

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

11.35 The `enumxt*` 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 ($\langle number \rangle$).

11.35.1 Functions for item box width

_enumext_starred_columns_set_vii:

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

```

2707 \cs_new_protected:Nn \__enumext_starred_columns_set_vii:
2708 {
2709   \dim_compare:nNnT { \l__enumext_columns_sep_vii_dim } = { \c_zero_dim }
2710   {
2711     \dim_set:Nn \l__enumext_columns_sep_vii_dim
2712     {
2713       ( \l__enumext_labelwidth_vii_dim + \l__enumext_labelsep_vii_dim )
2714       / \l__enumext_columns_vii_int
2715     }
2716   }
2717   \int_set:Nn \l__enumext_tmpa_vii_int { \l__enumext_columns_vii_int - \c_one_int }
2718   \dim_set:Nn \l__enumext_item_width_vii_dim
2719   {
2720     ( \linewidth - \l__enumext_columns_sep_vii_dim * \l__enumext_tmpa_vii_int )
2721     / \l__enumext_columns_vii_int - \l__enumext_labelwidth_vii_dim
2722     - \l__enumext_labelsep_vii_dim
2723   }
2724   \dim_zero_new:N \itemwidth
2725 }

```

(End of definition for _enumext_starred_columns_set_vii:.)

_enumext_starred_joined_item_vii:n

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

```

2726 \cs_new_protected:Npn \_enumext_starred_joined_item_vii:n #1
2727 {
2728   \int_set:Nn \l__enumext_joined_item_vii_int {#1}
2729   \int_compare:nNnT { \l__enumext_joined_item_vii_int } > { \l__enumext_columns_vii_int }
2730   {
2731     \msg_warning:nnee { enumext } { item-joined }
2732     { \int_use:N \l__enumext_joined_item_vii_int }
2733     { \int_use:N \l__enumext_columns_vii_int }
2734     \int_set:Nn \l__enumext_joined_item_vii_int
2735     {
2736       \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + \c_one_int
2737     }
2738   }
2739   \int_compare:nNnT
2740   { \l__enumext_joined_item_vii_int }
2741   >
2742   { \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + \c_one_int }
2743   {
2744     \msg_warning:nnee { enumext } { item-joined-columns }
2745     { \int_use:N \l__enumext_joined_item_vii_int }
2746     {
2747       \int_eval:n
2748       { \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + \c_one_int }
2749     }
2750     \int_set:Nn \l__enumext_joined_item_vii_int
2751     {
2752       \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + \c_one_int
2753     }
2754   }

```

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

```

2755   \int_compare:nNnTF { \l__enumext_joined_item_vii_int } > { \c_one_int }
2756   {
2757     \int_set_eq:NN \l__enumext_joined_item_aux_vii_int \l__enumext_joined_item_vii_int
2758     \int_decr:N \l__enumext_joined_item_aux_vii_int
2759     \int_add:Nn \l__enumext_item_column_pos_vii_int { \l__enumext_joined_item_aux_vii_int }
2760     \int_gadd:Nn \g__enumext_item_count_all_vii_int { \l__enumext_joined_item_aux_vii_int }
2761     \dim_set:Nn \l__enumext_joined_width_vii_dim
2762     {
2763       \l__enumext_item_width_vii_dim * \l__enumext_joined_item_vii_int
2764       + ( \l__enumext_labelwidth_vii_dim + \l__enumext_labelsep_vii_dim
2765         + \l__enumext_columns_sep_vii_dim
2766         ) * \l__enumext_joined_item_aux_vii_int

```

```

2767     }
2768     \dim_set_eq:NN \itemwidth \l__enumext_joined_width_vii_dim
2769   }
2770   {
2771     \dim_set_eq:NN \l__enumext_joined_width_vii_dim \l__enumext_item_width_vii_dim
2772     \dim_set_eq:NN \itemwidth \l__enumext_item_width_vii_dim
2773   }
2774 }

```

(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`.

```

2775 \cs_new_protected:Nn \__enumext_start_mini_vii:
2776 {
2777   \dim_compare:nNnT { \l__enumext_minipage_right_vii_dim } > { \c_zero_dim }
2778   {
2779     \dim_set:Nn \l__enumext_minipage_left_vii_dim
2780     {
2781       \linewidth
2782       - \l__enumext_minipage_right_vii_dim
2783       - \l__enumext_minipage_hsep_vii_dim
2784     }
2785     \bool_set_true:N \l__enumext_minipage_active_vii_bool
2786     \dim_gset_eq:NN
2787       \g__enumext_minipage_right_vii_dim
2788       \l__enumext_minipage_right_vii_dim
2789     \__enumext_mini_addvspace_vii:
2790     \nointerlineskip\noindent
2791     \begin{__enumext_mini_env*}{ \l__enumext_minipage_left_vii_dim }
2792   }
2793 }

```

(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*”.

```

2794 \cs_new_protected:Nn \__enumext_stop_mini_vii:
2795 {
2796   \bool_if:NT \l__enumext_minipage_active_vii_bool
2797   {
2798     \end{__enumext_mini_env*}
2799     \hfill
2800     \bool_gset_true:N \g__enumext_minipage_active_vii_bool
2801   }
2802 }

```

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*”.

```

2803 \__enumext_after_env:nn {enumext*}
2804 {
2805   \bool_if:NT \g__enumext_minipage_active_vii_bool
2806   {
2807     \begin{__enumext_mini_env*}{ \g__enumext_minipage_right_vii_dim }
2808     \par\addvspace { \g__enumext_minipage_right_skip }
2809     \bool_if:NF \g__enumext_minipage_center_vii_bool
2810     {
2811       \centering
2812     }
2813     \tl_use:N \g__enumext_miniright_code_vii_tl % the code
2814     \end{__enumext_mini_env*}
2815     \par\addvspace{ \g__enumext_minipage_after_skip }
2816   }
2817   \bool_gset_false:N \g__enumext_minipage_active_vii_bool
2818   \bool_gset_true:N \g__enumext_minipage_center_vii_bool
2819   \tl_gclear:N \g__enumext_miniright_code_vii_tl
2820   \dim_gzero:N \g__enumext_minipage_right_vii_dim
2821 }

```

(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.

```

2822 \NewDocumentEnvironment{enumext*}{o }
2823 {
2824   \__enumext_safe_exec_vii:
2825   \__enumext_parse_keys_vii:n {#1}
2826   \__enumext_before_list_vii:
2827   \__enumext_start_store_level_vii:
2828   \__enumext_start_list:nn { }
2829   {
2830     \__enumext_list_arg_two_vii:
2831     \__enumext_before_keys_exec_vii:
2832   }
2833   \__enumext_starred_columns_set_vii:
2834   \item[] \scan_stop:
2835   \cs_set_eq:NN \__enumext_stop_item_tmp_vii: \noindent
2836   \cs_set_eq:NN \item \__enumext_start_item_tmp_vii:
2837 }
2838 {
2839   \__enumext_stop_item_tmp_vii:
2840   \__enumext_remove_extra_parsep_vii:
2841   \__enumext_stop_list:
2842   \__enumext_stop_store_level_vii:
2843   \__enumext_after_list_vii:
2844 }
```

(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`.

```

2845 \cs_new_protected:Nn \__enumext_safe_exec_vii:
2846 {
2847   \int_incr:N \l__enumext_level_h_int
2848   \int_compare:nNnT { \l__enumext_level_h_int } > { 1 }
2849   {
2850     \msg_error:nn { enumext } { nested }
2851   }
2852   \bool_set_true:N \l__enumext_starred_bool
2853   \bool_lazy_all:nT
2854   {
2855     { \bool_not_p:n { \l__enumext_standar_bool } }
2856     { \int_compare_p:nNn { \l__enumext_level_int } = { \c_zero_int } }
2857   }
2858   {
2859     \bool_gset_true:N \g__enumext_starred_bool
2860   }
2861 }
```

(End of definition for `__enumext_safe_exec_vii:`.)

`__enumext_parse_keys_vii:n` Parse [`<key = val>`] for `enumext*`. If the variable `\l__enumext_store_active_bool` is true it will call the function `__enumext_parse_store_keys_vii:n` and reprocess the keys to pass them to the storage sequence.

```

2862 \cs_new_protected:Npn \__enumext_parse_keys_vii:n #1
2863 {
2864   \tl_if_novalue:nF {#1}
2865   {
2866     \keys_set:nn { enumext / enumext* } {#1}
2867     \bool_if:NT \l__enumext_store_active_bool
2868     {
2869       \__enumext_parse_store_keys_vii:n {#1}
2870     }
2871   }
2872 }
```

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

__enumext_parse_store_keys_vii:n

The function __enumext_parse_store_keys_vii:n searches for the values of the `columns` and `columns-sep` keys in the optional argument in `enumext*` environment as long as the starred versions of the `columns*` and `columns-sep*` keys are not active. The captured values are stored in the variable \l__enumext_store_opt_vii_tl which is used by the function __enumext_store_level_open_vii:.

```

2873 \cs_new_protected:Npn \__enumext_parse_store_keys_vii:n #1
2874 {
2875   \bool_if:NF \l__enumext_store_columns_vii_bool
2876   {
2877     \regex_match:nnT { \b columns\b } {#1}
2878     {
2879       \int_set_eq:NN
2880         \l__enumext_store_columns_vii_int
2881         \l__enumext_columns_vii_int
2882       \tl_put_right:Ne \l__enumext_store_opt_vii_tl
2883         {
2884           columns = \exp_not:V \l__enumext_store_columns_vii_int ,
2885         }
2886     }
2887   }
2888   \bool_if:NF \l__enumext_store_columns_sep_vii_bool
2889   {
2890     \regex_match:nnT { \b columns-sep\b } {#1}
2891     {
2892       \dim_set_eq:NN
2893         \l__enumext_store_columns_sep_vii_dim
2894         \l__enumext_columns_sep_vii_dim
2895       \tl_put_right:Ne \l__enumext_store_opt_vii_tl
2896         {
2897           columns-sep = \exp_not:V \l__enumext_store_columns_sep_vii_dim,
2898         }
2899     }
2900   }
2901 }

```

(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, then call the function __enumext_start_mini_vii: handle by `mini-env`.

```

2902 \cs_new_protected:Nn \__enumext_before_list_vii:
2903 {
2904   \__enumext_vspace_above_vii:
2905   \__enumext_before_args_exec_vii:
2906   \__enumext_start_mini_vii:
2907 }

```

(End of definition for __enumext_before_list_vii:.)

__enumext_after_list_vii:

The function __enumext_after_list: firsts 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.

```

2908 \cs_new_protected:Nn \__enumext_after_list_vii:
2909 {
2910
2911   \__enumext_stop_mini_vii:
2912   \__enumext_after_stop_list_vii:
2913   \__enumext_vspace_below_vii:
2914   \int_gset_eq:NN \g__enumext_resume_vii_int \value{enumXvii}
2915   \int_if_exist:cT { g__enumext_resume_ \l__enumext_store_name_tl _int }
2916   {
2917     \int_gset_eq:cN
2918       { g__enumext_resume_ \l__enumext_store_name_tl _int }
2919     { \value{enumXvii} }
2920   }
2921   \bool_lazy_and:nnT { \g__enumext_starred_bool } { \l__enumext_check_ans_bool }
2922   {

```

```

2923         \bool_gset_true:N \g__enumext_check_ans_show_h_bool
2924         \tl_gset:NV \g__enumext_store_name_tl \l__enumext_store_name_tl
2925     }
2926     \bool_gset_false:N \g__enumext_starred_bool
2927     \bool_set_false:N \l__enumext_starred_bool
2928 }

```

(End of definition for `__enumext_after_list_vii:`)

```

\__enumext_start_store_level_vii:
\__enumext_stop_store_level_vii:

```

The `__enumext_start_store_level_vii:` and `__enumext_stop_store_level_vii:` functions activate the level saving mechanism for storage in *(sequence)* of the `\anskey` command if `enumext*` are nested in `enumext`.

```

2929 \cs_new_protected:Nn \__enumext_start_store_level_vii:
2930 {
2931     \bool_if:NT \l__enumext_store_active_bool
2932     {
2933         \int_compare:nNt { \l__enumext_level_int } > { \c_zero_int }
2934         {
2935             \__enumext_store_level_open_vii:
2936         }
2937     }
2938 }
2939 \cs_new_protected:Nn \__enumext_stop_store_level_vii:
2940 {
2941     \bool_if:NT \l__enumext_store_active_bool
2942     {
2943         \int_compare:nNt { \l__enumext_level_int } > { \c_zero_int }
2944         {
2945             \__enumext_store_level_close_vii:
2946         }
2947     }
2948 }

```

(End of definition for `__enumext_start_store_level_vii:` and `__enumext_stop_store_level_vii:`)

11.35.2 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`.

```

2949 \cs_new_protected_nopar:Nn \__enumext_start_item_tmp_vii:
2950 {
2951     \__enumext_stop_item_tmp_vii:
2952     \int_incr:N \l__enumext_item_column_pos_vii_int
2953     \int_gincr:N \g__enumext_item_count_all_vii_int
2954     \__enumext_item_peek_args_vii:
2955 }

```

(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).

```

2956 \cs_new_protected:Nn \__enumext_item_peek_args_vii:
2957 {
2958     \peek_meaning:NTF (
2959         { \__enumext_joined_item_vii:w }
2960         { \__enumext_joined_item_vii:w (1) }
2961     }

```

(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`.

```

2962 \cs_new_protected:Npn \__enumext_joined_item_vii:w (#1)
2963 {

```

```

2964     \__enumext_starred_joined_item_vii:n {#1}
2965     \peek_meaning_remove:NTF *
2966     { \__enumext_starred_item_vii:w }
2967     { \__enumext_standard_item_vii:w }
2968 }

```

(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].

```

2969 \cs_new_protected:Npn \__enumext_standard_item_vii:w
2970 {
2971     \bool_set_false:N \l__enumext_item_starred_vii_bool
2972     \peek_meaning:NTF [
2973     {
2974         \bool_set_eq:NN
2975         \l__enumext_wrap_label_vii_bool
2976         \l__enumext_wrap_label_opt_vii_bool
2977         \__enumext_start_item_vii:w
2978     }
2979     {
2980         \bool_set_true:N \l__enumext_wrap_label_vii_bool
2981         \legacy_if_set_true:n { @noitemarg }
2982         \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
2983     }
2984 }

```

(End of definition for __enumext_standard_item_vii:w.)

__enumext_starred_item_vii:w

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

__enumext_starred_item_vii_aux_i:w
__enumext_starred_item_vii_aux_ii:w
__enumext_starred_item_vii_aux_iii:w

```

2985 \cs_new_protected:Npn \__enumext_starred_item_vii:w
2986 {
2987     \bool_set_true:N \l__enumext_item_starred_vii_bool
2988     \bool_set_true:N \l__enumext_wrap_label_vii_bool
2989     \peek_meaning:NTF [
2990     { \__enumext_starred_item_vii_aux_i:w }
2991     { \__enumext_starred_item_vii_aux_ii:w }
2992 }
2993 \cs_new_protected:Npn \__enumext_starred_item_vii_aux_i:w [#1]
2994 {
2995     \tl_gset:Nn \g__enumext_item_symbol_aux_vii_tl {#1}
2996     \__enumext_starred_item_vii_aux_ii:w
2997 }
2998 \cs_new_protected:Npn \__enumext_starred_item_vii_aux_ii:w
2999 {
3000     \peek_meaning:NTF [
3001     { \__enumext_starred_item_vii_aux_iii:w }
3002     {
3003         \dim_set_eq:NN
3004         \l__enumext_item_symbol_sep_vii_dim
3005         \l__enumext_labelsep_vii_dim
3006         \legacy_if_set_true:n { @noitemarg }
3007         \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
3008     }
3009 }
3010 \cs_new_protected:Npn \__enumext_starred_item_vii_aux_iii:w [#1]
3011 {
3012     \dim_set:Nn \l__enumext_item_symbol_sep_vii_dim {#1}
3013     \legacy_if_set_true:n { @noitemarg }
3014     \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
3015 }

```

(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.

```

3016 \cs_new_protected_nopar:Npn \__enumext_start_item_vii:w [#1]
3017 {
3018   \cs_set_eq:NN \__enumext_stop_item_tmp_vii: \__enumext_stop_item_vii:
3019   \legacy_if:nT { @noitemarg }
3020   {
3021     \legacy_if_set_false:n { @noitemarg }
3022     \legacy_if:nT { @nmbrlist }
3023     {
3024       \bool_if:NT \__enumext_hyperref_bool
3025       {
3026         \legacy_if_set_true:n { @hyper@item }
3027       }
3028       \refstepcounter{enumXvii}
3029       % code for check-ans
3030       \bool_if:NT \__enumext_check_ans_bool
3031       {
3032         % If true |no-store| key => nested in |enumext|
3033         \bool_if:NTF \__enumext_store_ans_bool
3034         {
3035           \int_gadd:cn { g__enumext_count_item_ \__enumext_level: _int }
3036           { \int_use:c { g__enumext_count_level_ \__enumext_level: _int } + 1 }
3037         }
3038         {
3039           \int_gincr:N \g__enumext_count_item_all_int
3040           \int_gincr:N \g__enumext_count_level_vii_int
3041         }
3042       }
3043     }
3044   }

```

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 `__enumext_footnotes_key_bool` is false, we will redefine the command `\footnote`, followed by printing the $\langle symbol \rangle$ defined for `\item*` if it is present and open a new group inside which we execute `font` key next to `\item` and the keys `wrap-label`, `wrap-label*`, `align`, close the group and execute the key `labelsep` and then the key `first`. Finally we open the `minipage` environment and execute the `listparindent` key which will be equal to `\parindent`, the `parsep` key which will be equal to `\parskip` and the `itemindent` key.

```

3045 \group_begin:
3046 \lrbox{ \__enumext_item_text_vii_box }
3047 \bool_if:NF \__enumext_footnotes_key_bool
3048 {
3049   \__enumext_renew_footnote:
3050 }
3051 \bool_if:NT \__enumext_item_starred_vii_bool
3052 {
3053   \tl_if_blank:VT \g__enumext_item_symbol_aux_vii_tl
3054   {
3055     \tl_gset_eq:NN
3056     \g__enumext_item_symbol_aux_vii_tl \__enumext_item_symbol_vii_tl
3057   }
3058   \mode_leave_vertical:
3059   \skip_horizontal:n { -\__enumext_item_symbol_sep_vii_dim }
3060   \makebox[ \opt ][ r ]{ \g__enumext_item_symbol_aux_vii_tl }
3061   \skip_horizontal:N \__enumext_item_symbol_sep_vii_dim
3062   \tl_gclear:N \g__enumext_item_symbol_aux_vii_tl
3063 }
3064 \group_begin:
3065 \tl_use:N \__enumext_label_font_style_vii_tl
3066 \bool_if:NTF \__enumext_wrap_label_vii_bool
3067 {
3068   \makebox[ \__enumext_labelwidth_vii_dim ][ \__enumext_align_label_vii_str ]

```

```

3069         { \__enumext_wrapper_label_vii:n {#1} }
3070     }
3071     {
3072         \makebox[ \__enumext_labelwidth_vii_dim ][ \__enumext_align_label_vii_str ]{ #1 }
3073     }
3074     \group_end:
3075     \skip_horizontal:N \__enumext_labelsep_vii_dim
3076     \tl_use:N \__enumext_after_list_args_vii_tl
3077     \__enumext_minipage:w [ t ]{ \__enumext_joined_width_vii_dim }
3078     \skip_set_eq:NN \parindent \__enumext_listparindent_vii_dim
3079     \skip_set_eq:NN \parskip \__enumext_parsep_vii_skip
3080     \tl_use:N \__enumext_fake_item_indent_vii_tl
3081 }

```

(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.

```

3082 \cs_new_protected_nopar:Nn \__enumext_stop_item_vii:
3083 {
3084     \__enumext_endminipage:
3085     \endlrbox
3086     \group_end:
3087     \box_set_wd:Nn \__enumext_item_text_vii_box
3088     {
3089         \__enumext_joined_width_vii_dim
3090         + \__enumext_labelwidth_vii_dim
3091         + \__enumext_labelsep_vii_dim
3092     }
3093     \int_set:Nn \hbadness { 10000 }
3094     \box_use:N \__enumext_item_text_vii_box
3095     \bool_if:NF \__enumext_footnotes_key_bool
3096     {
3097         \__enumext_print_footnote:
3098     }
3099     \int_compare:nNnTF { \__enumext_item_column_pos_vii_int } = { \__enumext_columns_vii_int }
3100     {
3101         \par\noindent
3102         \int_zero:N \__enumext_item_column_pos_vii_int
3103     }
3104     { \hspace{ \__enumext_columns_sep_vii_dim } }
3105 }

```

(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.

```

3106 \cs_new_protected:Nn \__enumext_remove_extra_parsep_vii:
3107 {
3108     \int_compare:nNnT
3109     {
3110         \int_mod:nn { \__enumext_item_count_all_vii_int } { \__enumext_columns_vii_int }
3111     }
3112     =
3113     { \c_zero_int }
3114     {
3115         \par
3116         \vspace{ -\__enumext_itemsep_vii_skip }
3117         \int_gzero:N \__enumext_item_count_all_vii_int
3118     }
3119 }

```

(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.

```

3120 \__enumext_after_env:nn {enumext*}
3121 {
3122     \bool_if:NT \__enumext_check_ans_show_h_bool
3123     {

```

```

3124         \int_compare:nNnT { \l__enumext_level_int } = { 0 }
3125         {
3126             \__enumext_check_ans_active_vii:
3127         }
3128     }
3129     \bool_gset_false:N \g__enumext_check_ans_show_h_bool
3130     \tl_gclear:N \g__enumext_store_name_tl
3131 }

```

11.36 The command \getkeyans

`\getkeyans` The `\getkeyans` command takes a mandatory argument of the form $\langle \text{store name} : \text{position} \rangle$. Retrieve a “single” content stored by `\anskey`, `\anspic*` and `\item*` from $\langle \text{prop list} \rangle$ defined by `save-ans` key.

```

3132 \NewDocumentCommand \getkeyans { m }
3133 {
3134     \exp_args:Ne \__enumext_getkeyans_aux:n
3135     { \tl_to_str:e { \text_expand:n {#1} } }
3136 }

```

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

`__enumext_getkeyans_aux:n` The internal function `__enumext_getkeyans_aux:n` is in charge of *splitting* the $\langle \text{argument} \rangle$ using “:”. If “:” is omitted it will return an error.

```

3137 \cs_new_protected:Npn \__enumext_getkeyans_aux:n #1
3138 {
3139     \str_if_in:nnTF {#1} { : }
3140     {
3141         \use:e
3142         {
3143             \cs_set:Npn \exp_not:N \__enumext_tmp:w #1 \c_colon_str ##2 \scan_stop:
3144             { {##1} {##2} }
3145         }
3146         \exp_after:wN \__enumext_getkeyans:nn \__enumext_tmp:w #1 \scan_stop:
3147     }
3148     { \msg_error:nnn { enumext } { missing-colon } {#1} }
3149 }

```

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

`__enumext_getkeyans:nn` The internal function `__enumext_getkeyans:nn` will check for the existence of the $\langle \text{prop list} \rangle$, if it does not exist it will return an error message, then it will fetch the content specified by the second $\langle \text{argument} \rangle$ from $\langle \text{prop list} \rangle$.

```

3150 \cs_new_protected:Npn \__enumext_getkeyans:nn #1 #2
3151 {
3152     \prop_if_exist:cF { g__enumext_#1_prop }
3153     { \msg_error:nnn { enumext } { undefined-storage-anskey } {#1} }
3154     \group_begin:
3155         \prop_item:cn { g__enumext_#1_prop }{#2}
3156     \group_end:
3157 }

```

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

11.37 The command \printkeyans

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

```

3158 \keys_define:nn { keyanskey / print }
3159 {
3160     level-1 .code:n      = \tl_put_right:Nn \l__enumext_print_keyans_i_tl
3161                         {
3162                             \setenumext[level,1] {#1} \setenumext[print,1] {#1}
3163                         },
3164     level-1 .initial:n   = { label=\arabic*. , nosep, columns=2, first=\small, font=\small },
3165     level-2 .code:n      = \tl_put_right:Nn \l__enumext_print_keyans_ii_tl
3166                         {
3167                             \setenumext[level,2] {#1} \setenumext[print,2] {#1}
3168                         },
3169     level-2 .initial:n   = { nosep, label=(\alph*), first=\small, font=\small },

```

```

3170   level-3 .code:n      = \tl_put_right:Nn \l__enumext_print_keyans_iii_tl
3171                       {
3172                         \setenumext[level,3] {#1} \setenumext[print,3] {#1}
3173                       },
3174   level-3 .initial:n   = { nosep, label=\roman*., first=\small, font=\small },
3175   level-4 .code:n      = \tl_put_right:Nn \l__enumext_print_keyans_iv_tl
3176                       {
3177                         \setenumext[level,4] {#1} \setenumext[print,4] {#1}
3178                       },
3179   level-4 .initial:n   = { nosep, label=\Alph*., first=\small, font=\small },
3180   level-* .code:n      = \tl_put_right:Nn \l__enumext_print_keyans_vii_tl % starred
3181                       {
3182                         \setenumext[enumext*] {#1} %\setenumext[print,*] {#1}
3183                       },
3184   level-* .initial:n   = { label=\arabic*., nosep, columns=2, first=\small, font=\small },
3185   }

```

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

```

3186 \NewDocumentCommand \printkeyans { s O{} m }
3187 {
3188   \group_begin:
3189     \tl_use:N \l__enumext_print_keyans_i_tl
3190     \tl_use:N \l__enumext_print_keyans_ii_tl
3191     \tl_use:N \l__enumext_print_keyans_iii_tl
3192     \tl_use:N \l__enumext_print_keyans_iv_tl
3193     \tl_use:N \l__enumext_print_keyans_vii_tl
3194     \__enumext_printkeyans:nnn { #1 } { #2 } { #3 }
3195   \group_end:
3196 }

```

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

__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

```

3197 \cs_new_protected:Npn \__enumext_printkeyans:nnn #1 #2 #3
3198 {
3199   \seq_if_exist:cTF { g__enumext_#3_seq }
3200   {
3201     \seq_if_empty:cF { g__enumext_#3_seq }
3202     {
3203       \bool_if:nTF {#1}
3204       {
3205         \begin{enumext*}[#2]
3206         \seq_map_inline:cn { g__enumext_#3_seq } { ##1 }
3207         \end{enumext*}
3208       }
3209       {
3210         \begin{enumext}[#2]
3211         \seq_map_inline:cn { g__enumext_#3_seq } { ##1 }
3212         \end{enumext}
3213       }
3214     }
3215   }
3216   {
3217     \msg_error:nnn { enumext } { undefined-storage-anskey } {#3}
3218   }
3219 }

```

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

11.38 The command *\setenumext*

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

```

3220 \keys_define:nn { enumext / meta-families }
3221 {
3222   level-1 .code:n = { \keys_set:nn { enumext / level-1 } {#1} },
3223   level-2 .code:n = { \keys_set:nn { enumext / level-2 } {#1} },

```



```

3224   level-3   .code:n = { \keys_set:nn { enumext / level-3 } {#1} } ,
3225   level-4   .code:n = { \keys_set:nn { enumext / level-4 } {#1} } ,
3226   keyans    .code:n = { \keys_set:nn { enumext / keyans   } {#1} } ,
3227   enumext*   .code:n = { \keys_set:nn { enumext / enumext* } {#1} } ,
3228   keyans*    .code:n = { \keys_set:nn { enumext / keyans*  } {#1} } ,
3229   print-1   .code:n = { \keys_set:nn { keyanskey / print } { level-1 = {#1} } } ,
3230   print-2   .code:n = { \keys_set:nn { keyanskey / print } { level-2 = {#1} } } ,
3231   print-3   .code:n = { \keys_set:nn { keyanskey / print } { level-3 = {#1} } } ,
3232   print-4   .code:n = { \keys_set:nn { keyanskey / print } { level-4 = {#1} } } ,
3233   print-*   .code:n = { \keys_set:nn { keyanskey / print } { level-* = {#1} } } ,
3234   unknown   .code:n = { \msg_error:nn { enumext } { unknown-key-family } } ,
3235 }

```

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

```

3236 \seq_const_from_clist:Nn \c__enumext_all_families_seq
3237 {
3238   level-1 , level-2 , level-3 , level-4 , keyans , enumext* ,
3239   keyans* , print-1 , print-2 , print-3 , print-4 , print-* ,
3240 }

```

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

```

3241 \NewDocumentCommand \setenumext { o +m }
3242 {
3243   \tl_if_novalue:nTF {#1}
3244   {
3245     \seq_map_inline:Nn \c__enumext_all_families_seq
3246     {
3247       {
3248         \seq_clear:N \l__enumext_setkey_tmpa_seq
3249         \seq_set_from_clist:Nn \l__enumext_setkey_tmpb_seq {#1}
3250         \int_set:Nn \l__enumext_setkey_tmpa_int
3251         {
3252           \seq_count:N \l__enumext_setkey_tmpb_seq
3253         }
3254         \int_compare:nNnTF { \l__enumext_setkey_tmpa_int } > { 1 }
3255         {
3256           \seq_pop_left:NN \l__enumext_setkey_tmpb_seq \l__enumext_setkey_tmpa_tl
3257           \seq_map_function:NN \l__enumext_setkey_tmpb_seq \l__enumext_set_parse:n
3258           \seq_set_map_e:NNn \l__enumext_setkey_tmpa_seq \l__enumext_setkey_tmpa_seq
3259           {
3260             \tl_use:N \l__enumext_setkey_tmpa_tl - ##1
3261           }
3262         }
3263         {
3264           \seq_put_right:Ne \l__enumext_setkey_tmpa_seq { \tl_trim_spaces:n {#1} }
3265         }
3266         \seq_if_empty:NTF \l__enumext_setkey_tmpa_seq
3267         { \seq_map_inline:Nn \c__enumext_all_families_seq }
3268         { \seq_map_inline:Nn \l__enumext_setkey_tmpa_seq }
3269       }
3270     }
3271     \keys_set:nn { enumext / meta-families } { ##1 = {#2} }
3272   }
3273 }

```

(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.

```

3274 \cs_new_protected:Npn \__enumext_set_parse:n #1
3275 {
3276   \tl_set:Ne \l__enumext_setkey_tmpb_tl { \tl_trim_spaces:n {#1} }
3277   \int_step_inline:nnn { 0 } { 4 } %<- max level
3278   { \tl_remove_all:Nn \l__enumext_setkey_tmpb_tl {##1} }
3279   \tl_if_empty:NTF \l__enumext_setkey_tmpb_tl
3280   {
3281     \seq_put_right:Ne \l__enumext_setkey_tmpa_seq
3282     { \tl_trim_spaces:n {#1} }
3283   }
3284   { \__enumext_set_error:nn {#1} { } }
3285 }
3286 \cs_new_protected:Npn \__enumext_set_error:nn #1#2
3287 { \msg_error:nnn { enumext } { invalid-key } {#1} {#2} }

```

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

11.39 Messages

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

```
3288 \msg_new:nnn { enumext } { counters }
3289 {
3290   The ~ counter ~ '#1' ~ is ~ already ~ defined ~ by ~ some ~ \\
3291   package ~ or ~ macro, ~ it ~ cannot ~ be ~ continued.
3292 }
```

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

```
3293 \msg_new:nnn { enumext } { invalid-key }
3294 {
3295   The ~ key ~ '#1' ~ is ~ not ~ know ~ the ~ level ~ #2.
3296 }
3297 \msg_new:nnn { enumext } { unknown-key-family }
3298 {
3299   Unknown~key~family~`\l_keys_key_str'~for~enumext.
3300 }
```

Messages used in length calculation.

```
3301 \msg_new:nnn { enumext } { width-negative }
3302 {
3303   Ignoring ~ negative ~ value ~ '#1=#2' ~ \msg_line_context:.\
3304   The ~ key ~ '#1'~ accepts ~ values ~ >= ~ 0pt.
3305 }
3306 \msg_new:nnn { enumext } { width-zero }
3307 {
3308   Invalid ~ '#1=#2' ~ \msg_line_context:.\
3309   The ~ key ~ '#1'~ accepts ~ values ~ > ~ 0pt.
3310 }
```

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

```
3311 \msg_new:nnn { enumext } { list-lengths }
3312 {
3313   **** ~ Lengths ~ used ~ by ~ 'enumext' ~ level ~ '#2' ~ \msg_line_context:~\c_space_tl ****\
3314   \__enumext_show_length:nnn { dim } { labelsep } {#1}
3315   \__enumext_show_length:nnn { dim } { labelwidth } {#1}
3316   \__enumext_show_length:nnn { dim } { itemindent } {#1}
3317   \__enumext_show_length:nnn { dim } { leftmargin } {#1}
3318   \__enumext_show_length:nnn { dim } { rightmargin } {#1}
3319   \__enumext_show_length:nnn { dim } { listparindent } {#1}
3320   \__enumext_show_length:nnn { skip } { topsep } {#1}
3321   \__enumext_show_length:nnn { skip } { parsep } {#1}
3322   \__enumext_show_length:nnn { skip } { partopsep } {#1}
3323   \__enumext_show_length:nnn { skip } { itemsep } {#1}
3324   ****~
3325 }
```

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

```
3326 \msg_new:nnn { enumext } { list-lengths-not-nested }
3327 {
3328   **** ~ Lengths ~ used ~ by ~ '#2' ~ environment ~ \msg_line_context:~\c_space_tl ****\
3329   \__enumext_show_length:nnn { dim } { labelsep } {#1}
3330   \__enumext_show_length:nnn { dim } { labelwidth } {#1}
3331   \__enumext_show_length:nnn { dim } { itemindent } {#1}
3332   \__enumext_show_length:nnn { dim } { leftmargin } {#1}
3333   \__enumext_show_length:nnn { dim } { rightmargin } {#1}
3334   \__enumext_show_length:nnn { dim } { listparindent } {#1}
3335   \__enumext_show_length:nnn { skip } { topsep } {#1}
3336   \__enumext_show_length:nnn { skip } { parsep } {#1}
3337   \__enumext_show_length:nnn { skip } { partopsep } {#1}
3338   \__enumext_show_length:nnn { skip } { itemsep } {#1}
3339   ****~
3340 }
```

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

```
3341 \msg_new:nnn { enumext } { items-same-answer }
3342 {
3343   *****~Checking~answers~on~'#1'~OK~*****\
3344   **~ All ~ items ~ stored ~ in ~ sequence ~ '#1' ~ have ~ an ~ answer. \
3345   *****~
3346   \prg_replicate:nn { 7 + \str_count:n {#1} } { * }
```

```

3347     }
3348     \msg_new:nnn { enumext } { item-different-answer }
3349     {
3350         Number ~ of ~ items ~ different ~ of ~ number ~ of ~
3351         answer ~ in ~ sequence ~ '#1' ~ closed ~ \msg_line_context:.
3352     }

```

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

```

3353     \msg_new:nnn { enumext } { missing-starred }
3354     {
3355         Missing ~ '\c_backslash_str #1*' ~ in ~ '#2' ~ \msg_line_context:.
3356     }

```

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

```

3357     \msg_new:nnn { enumext } { list-too-deep }
3358     {
3359         Too ~ deep ~ nesting ~ for ~ 'enumext' ~ \msg_line_context:~ \\
3360         The ~ maximum ~ level ~ of ~ nesting ~ is ~ 4.
3361     }

```

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

```

3362     \msg_new:nnn { enumext } { anskey-wrong-place }
3363     {
3364         Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:~ \\
3365         '\c_backslash_str #1' ~ works ~ in ~ the ~ environment ~ '#2'.
3366     }
3367     \msg_new:nnn { enumext } { anspic-wrong-place }
3368     {
3369         Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:~ \\
3370         '\c_backslash_str #1' ~ works ~ in ~ the ~ environment ~ '#2'.
3371     }
3372     \msg_new:nnn { enumext } { command-wrong-place }
3373     {
3374         Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:~ \\
3375         '\c_backslash_str #1' ~ works ~ outside ~ the ~ environment ~ '#2'.
3376     }

```

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

```

3377     \msg_new:nnn { enumext } { keyans-nested }
3378     {
3379         The ~ environment ~ 'keyans' ~ can't ~ be ~ nested ~ \msg_line_context:.
3380     }
3381     \msg_new:nnn { enumext } { keyans-wrong-level }
3382     {
3383         Wrong ~ level ~ position ~ for ~ 'keyans' ~ \msg_line_context:~ \\
3384         The ~ environment ~ 'keyans' ~ can ~ only ~ be ~ in ~ the ~ first ~ level.
3385     }
3386     \msg_new:nnn { enumext } { wrong-place }
3387     {
3388         Wrong ~ place ~ for ~ '#1' ~ environment ~ \msg_line_context:~ \\
3389         '#1' ~ is ~ only ~ found ~ with ~ '#2' ~ in ~ 'enumext'.
3390     }
3391     \msg_new:nnn { enumext } { keyanspic-nested }
3392     {
3393         The ~ environment ~ 'keyanspic' ~ can't ~ be ~ nested ~ \msg_line_context:~.
3394     }
3395     \msg_new:nnn { enumext } { keyanspic-wrong-level }
3396     {
3397         Wrong ~ level ~ position ~ for ~ 'keyanspic' ~ \msg_line_context:~ \\
3398         The ~ environment ~ 'keyans' ~ can ~ only ~ be ~ in ~ the ~ first ~ level.
3399     }

```

Messages used by `\getkeyans` command.

```

3400     \msg_new:nnn { enumext } { undefined-storage-anskey }
3401     {
3402         Storage ~ named ~ '#1' ~ is ~ not ~ defined ~ \msg_line_context:.
3403     }

```

Messages used by `\miniright` command.

```

3404     \msg_new:nnn { enumext } { missing-miniright }
3405     {
3406         Missing ~ '\c_backslash_str miniright' ~ in ~ \msg_line_context:~ \\
3407         The ~ key ~ 'mini-env' ~ need ~ '\c_backslash_str miniright'.
3408     }

```

```

3409 \msg_new:nnn { enumext } { wrong-miniright-place }
3410 {
3411   Wrong ~ place ~ for ~ '\c_backslash_str miniright' ~ \msg_line_context:~ \\
3412   Works ~ in ~ 'enumext' ~ and ~ 'keyans' ~ with ~ key ~ 'mini-env'.
3413 }
3414 \msg_new:nnn { enumext } { wrong-miniright-use }
3415 {
3416   Wrong ~ use ~ for ~ '\c_backslash_str miniright' ~ \msg_line_context:~ \\
3417   '\c_backslash_str miniright' ~ need ~ a ~ key ~ 'mini-env'.
3418 }

```

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

```

3419 \msg_new:nnn { enumext } { nested }
3420 {
3421   The ~ starred ~ environment ~ can't ~ be ~ nested ~ \msg_line_context:.
3422 }
3423 \msg_new:nnn { enumext } { item-joined }
3424 {
3425   Items ~ joined ~ (#1) ~ > ~ #2 ~ columns ~ \msg_line_context:.
3426 }
3427 \msg_new:nnn { enumext } { item-joined-columns }
3428 {
3429   Not ~ space ~ to ~ join ~ items ~ (#1) ~ > ~ #2 ~ \msg_line_context:.
3430 }

```

11.40 Finish package

Finish package implementation.

```

3431 \file_input_stop:
3432 </package>

```

12 Index of Implementation

The italic numbers denote the pages where the corresponding entry is described, the numbers underlined and all others indicate the line on which they are implemented in the package code.

Symbols

*

.....

383

\+

.....

193

\-

.....

193

\\

201, 2613, 3290, 3303, 3308, 3313, 3328, 3343, 3344, 3359, 3364, 3369, 3374, 3383, 3388, 3397, 3406, 3411, 3416

A

above

.....

1194

above*

.....

1194

\addvspace

841, 869, 992, 1071, 1134, 1140, 1168, 1185, 2432, 2452, 2569, 2584, 2808, 2815

after

.....

679

align

.....

337

\Alph

.....

27, 31

\Alph

.....

289, 466, 484, 497, 3179

\alph

.....

27, 31

\alph

.....

290, 464, 3169

\anskey

.....

9, 54, 1633

\anspic

.....

11, 74, 75, 2591

\arabic

.....

27, 28

\arabic

.....

288, 463, 483, 3164, 3184

B

\b

.....

2323, 2336, 2877, 2890

\baselineskip

.....

39

\baselineskip

.....

1593, 1601

before

.....

679

before*

.....

679

below

.....

1194

below*

.....

1194

bool commands:

\bool_gset_false:N

.....

2475, 2817, 2926, 3129

\bool_gset_true:N

783, 2307, 2437, 2800, 2818, 2859, 2923

\bool_if:NTF

229, 241, 258, 1216, 1230, 1243, 1254, 1265, 1276, 1287, 1298, 1329, 1336, 1347, 1405, 1407, 1555, 1579, 1586, 1614, 1645, 1658, 1660, 1671, 1691, 1816, 1827, 1831, 1893, 1927, 1942, 1953, 2030, 2061, 2135, 2151, 2219, 2229, 2265, 2270, 2314, 2321, 2334, 2366, 2416, 2430, 2443, 2468, 2498, 2554, 2567, 2575, 2593, 2796, 2805, 2809, 2867, 2875, 2888, 2931, 2941, 3024, 3030, 3033, 3047, 3051, 3066, 3095, 3122

\bool_if:nTF

1169, 1186, 1699, 2073, 2107, 2171, 2614, 3203

\bool_if_p:N

.....

2208, 2255

\bool_lazy_all:nTF

1385, 1748, 1757, 1770, 1786, 2301, 2853

\bool_lazy_and:nnTF

1681, 1724, 1912, 2206, 2254, 2350, 2435, 2921

\bool_lazy_or:nnTF

.....

2619

\bool_new:N

18, 19, 20, 21, 22, 28, 30, 39, 60, 65, 66, 71, 72, 75, 91, 93, 95, 98, 99, 108, 109, 110, 121, 122, 147, 158, 160

\bool_not_p:n

1683, 1775, 1790, 2303, 2350, 2435, 2855

\bool_set_eq:NN

.....

2039, 2088, 2974

\bool_set_false:N

238, 1480, 1483, 1503, 1506, 2457, 2505, 2587, 2651, 2668, 2927, 2971

\bool_set_true:N

220, 224, 330, 607, 1200, 1205, 1324, 1343, 1354, 1479, 1482, 1502, 1505, 1515, 1522, 2035,

2066, 2084, 2096, 2300, 2354, 2359, 2385, 2503, 2529, 2785, 2852, 2980, 2987, 2988

box commands:

\box_dp:N

888, 892, 896, 907, 911, 922, 931, 937, 947, 960, 966, 972, 1003, 1004, 1005, 1008, 1018, 1022, 1031, 1038, 1043, 1051, 1080, 1081, 1084, 1091, 1104, 1112, 1118, 1126, 2680

\box_new:N

.....

36, 153

\box_set_wd:Nn

.....

3087

\box_use:N

.....

3094

\box_wd:N

.....

296

C

\c

.....

383, 384, 507, 509, 521, 523

\cB

.....

384

\cE

.....

384

\centering

.....

1171, 1188, 2701, 2811

check-ans

.....

1356

Document class:

article

.....

32

clist commands:

\clist_const:Nn

.....

165

\clist_map_function:nN

.....

2688

\clist_map_inline:Nn

336, 549, 612, 678, 693, 774, 1210

\clist_map_inline:nn

27, 44, 50, 62, 74, 97, 120, 130, 144, 164, 361, 378, 617, 789, 1366, 1492, 1510, 1531, 1745, 1872, 1988, 2200, 2203, 2236, 2246, 2249, 2275

\columnbreak

.....

55

\columnbreak

.....

1685

columns

.....

758

columns*

.....

1511

columns-sep

.....

758

columns-sep*

.....

1511

\columnsep

.....

70, 73

\columnsep

.....

2410, 2551

\columnseprule

.....

70, 73

\columnseprule

.....

2414, 2553

Commands provide by enumext:

\anskey

21, 22, 47-49, 52, 54, 56-58, 60, 69, 82, 86, 87, 90

\anspic*

.....

21, 58-60, 75-77, 86, 87

\anspic

.....

52, 74-77, 90

\getkeyans

.....

52, 86, 90

\item*

.....

21, 52, 58-60, 62, 63, 83, 86, 87

\itemwidth

.....

78

\item

.....

62, 63, 78, 82-84

\miniright

.....

20, 36, 44, 45, 70, 71, 73, 74, 90

\printkeyans

.....

21, 52, 86

\setenumext

.....

21, 87-89

Counters defined by enumext:

enumXiii

.....

20, 26

enumXii

.....

20, 26

enumXiv

.....

20, 26

enumXi

.....

20, 26

enumXviii

.....

20, 26

enumXvii

.....

20, 26, 84

enumXvi

.....

20, 26

enumXv

.....

20, 26

©2024 by Pablo González L

92 / 102

cs commands:

`\cs_generate_variant:Nn` 298, 314, 513, 529, 1545, 1552, 1632, 2190, 2690
`\cs_if_exist:NTF` 268
`\cs_new:Nn` 178
`\cs_new:Npn` 182, 187, 197
`\cs_new_eq:NN` 205, 206, 207, 211, 212, 243, 244, 247, 248
`\cs_new_protected:Nn` . 216, 379, 399, 431, 694, 698, 702, 706, 710, 714, 718, 722, 726, 730, 734, 738, 742, 746, 750, 754, 790, 802, 826, 843, 854, 878, 953, 977, 994, 1056, 1073, 1095, 1130, 1136, 1211, 1225, 1239, 1250, 1261, 1272, 1283, 1294, 1334, 1345, 1403, 1415, 1438, 1553, 1577, 1584, 1612, 1619, 1736, 1863, 1993, 1997, 2016, 2069, 2103, 2119, 2129, 2145, 2295, 2348, 2364, 2371, 2394, 2424, 2441, 2496, 2519, 2537, 2562, 2573, 2610, 2653, 2666, 2686, 2691, 2707, 2775, 2794, 2845, 2902, 2908, 2929, 2939, 2956, 3106
`\cs_new_protected:Npn` 170, 174, 251, 266, 283, 293, 299, 387, 406, 500, 514, 1158, 1177, 1321, 1371, 1536, 1546, 1668, 1813, 1825, 1847, 1898, 1932, 1940, 2026, 2045, 2080, 2092, 2159, 2193, 2239, 2310, 2319, 2515, 2661, 2726, 2862, 2873, 2962, 2969, 2985, 2993, 2998, 3010, 3137, 3150, 3197, 3274, 3286
`\cs_new_protected_nopar:Nn` 2949, 3082
`\cs_new_protected_nopar:Npn` 3016
`\cs_set:Nn` 185, 1818
`\cs_set:Npn` 1746, 1784, 3143
`\cs_set_eq:NN` 184, 189, 2835, 2836, 3018
`\cs_set_protected:Nn` 618, 634, 646, 658
`\cs_set_protected:Npn` . 23, 37, 45, 57, 63, 87, 115, 126, 138, 145, 315, 337, 366, 447, 467, 530, 550, 594, 613, 670, 679, 758, 775, 1194, 1356, 1464, 1493, 1511, 1738, 1865, 1977, 2191, 2237
`\cs_to_str:N` 285, 308

D

`\d` 193
`\DeclareDocumentEnvironment` 871

dim commands:

`\dim_abs:n` 2164, 2169
`\dim_add:Nn` 2671
`\dim_compare:nNnTF` . 620, 636, 648, 660, 1160, 1179, 2161, 2166, 2172, 2178, 2180, 2182, 2376, 2399, 2523, 2541, 2663, 2709, 2777
`\dim_compare:nTF` 1709
`\dim_gset_eq:NN` 2786
`\dim_gzero:N` 2820
`\dim_new:N` .. 33, 40, 41, 42, 59, 94, 104, 154, 155, 161
`\dim_set:Nn` .. 296, 608, 1523, 2059, 2164, 2169, 2171, 2174, 2175, 2179, 2181, 2184, 2185, 2187, 2379, 2402, 2525, 2543, 2693, 2711, 2718, 2761, 2779, 3012
`\dim_set_eq:NN` 454, 474, 490, 494, 2054, 2202, 2248, 2338, 2410, 2551, 2768, 2771, 2772, 2892, 3003
`\dim_use:N` 621, 629, 1161, 1167, 1622, 1625, 1630, 2124, 2126, 2377, 2382, 2383, 2390, 2400, 2404, 2405, 2407
`\dim_zero:N` 2414, 2553, 2672, 2673, 2674
`\dim_zero_new:N` 2724
`\c_zero_dim` 623, 637, 649, 661, 1161, 1179, 1711, 2161, 2166, 2172, 2179, 2377, 2400, 2523, 2541, 2709, 2777

E

`\end` .. 1164, 1182, 1581, 1616, 2429, 2451, 2566, 2583, 2798, 2814, 3207, 3212
`\endlist` 24
`\endlist` 206

`\endlrbox` 3085
`\endminipage` 25
`\endminipage` 212
enumext 4, 2276
enumext internal commands:

`__enumext_add_pre_parsep:` ... 37, 800, 802, 802
`__enumext_after_args_exec:` . 35, 694, 706, 2288
`__enumext_after_args_exec_v:` . 35, 36, 710, 722, 2489
`__enumext_after_args_exec_vii:` ... 726, 750
`__enumext_after_args_exec_viii:` 754
`__enumext_after_env:n` 71
`__enumext_after_env:nn` .. 71, 85, 174, 174, 2466, 2803, 3120
`__enumext_after_hyperref:` ... 25, 214, 216, 216
`__enumext_after_list:` . 71, 81, 2293, 2441, 2441
`\l__enumext_after_list_args_v_tl` 724
`\l__enumext_after_list_args_vii_tl` 752, 3076
`\l__enumext_after_list_args_viii_tl` ... 756
`__enumext_after_list_v:` .. 74, 2494, 2573, 2573
`__enumext_after_list_vii:` ... 2843, 2908, 2908
`__enumext_after_star_env:nn` 79
`__enumext_after_stop_list:` ... 35, 36, 694, 702, 2455
`__enumext_after_stop_list_v:` 35, 710, 718, 2588
`\l__enumext_after_stop_list_v_tl` 720
`__enumext_after_stop_list_vii:` 726, 742, 2912
`\l__enumext_after_stop_list_vii_tl` ... 744
`__enumext_after_stop_list_viii:` 746
`\l__enumext_after_stop_list_viii_tl` ... 748
`\l__enumext_align_label_vii_str` .. 3068, 3072
`\l__enumext_align_label_X_str` 145
`\c__enumext_all_envs_clist` .. 165, 336, 549, 612, 678, 693, 774, 1210
`\c__enumext_all_families_seq` .. 88, 3236, 3245, 3267
`__enumext_anskey_wrapper:n` 1468, 1823
`__enumext_at_begin_document:n` .. 24, 170, 170, 203, 209
`__enumext_before_args_exec:` 34, 694, 694, 2374
`__enumext_before_args_exec_v:` .. 35, 710, 710, 2522
`__enumext_before_args_exec_vii:` .. 726, 726, 2905
`__enumext_before_args_exec_viii:` 730
`__enumext_before_keys_exec:` 35, 694, 698, 2286
`__enumext_before_keys_exec_v:` .. 35, 710, 714, 2487
`__enumext_before_keys_exec_vii` 726
`__enumext_before_keys_exec_vii:` 36, 734, 2831
`__enumext_before_keys_exec_viii:` .. 36, 738
`__enumext_before_list:` ... 69, 2280, 2371, 2371
`__enumext_before_list_v:` . 73, 2482, 2519, 2519
`__enumext_before_list_vii:` 81, 2826, 2902, 2902
`\l__enumext_before_no_starred_key_v_tl` 716
`\l__enumext_before_no_starred_key_vii-`
`tl` 736
`\l__enumext_before_no_starred_key_viii-`
`tl` 740
`\l__enumext_before_starred_key_v_tl` ... 712
`\l__enumext_before_starred_key_vii_tl` . 728
`\l__enumext_before_starred_key_viii_tl` 732
`__enumext_calc_hspace:NNNNNNN` 65, 2159, 2159, 2190, 2195, 2241

__enumext_check_ans_active: . . 50, [1415](#), [1415](#),
 [2472](#)
 __enumext_check_ans_active_vii: . [1415](#), [1438](#),
 [3126](#)
 \l__enumext_check_ans_bool . 62, [108](#), [1329](#), [1360](#),
 [1405](#), [1660](#), [1927](#), [2030](#), [2061](#), [2435](#), [2921](#), [3030](#)
 __enumext_check_ans_count: . 49, 69, [1403](#), [1403](#),
 [2375](#)
 __enumext_check_ans_int:n . . 47, 49, [1331](#), [1371](#),
 [1371](#)
 \g__enumext_check_ans_item_tl . . 60, [108](#), [1926](#),
 [1934](#), [1938](#)
 \g__enumext_check_ans_show_bool 71, [108](#), [2437](#),
 [2468](#), [2475](#)
 \g__enumext_check_ans_show_h_bool [108](#), [2923](#),
 [3122](#), [3129](#)
 \l__enumext_columns_sep_v_dim 2541, [2543](#), [2551](#)
 \l__enumext_columns_sep_vii_dim . . 2709, [2711](#),
 [2720](#), [2765](#), [2894](#), [3104](#)
 \l__enumext_columns_v_int 999, [2539](#), [2547](#), [2559](#),
 [2564](#)
 \l__enumext_columns_vii_int . . 2714, [2717](#), [2721](#),
 [2729](#), [2733](#), [2736](#), [2742](#), [2748](#), [2752](#), [2881](#), [3099](#), [3110](#)
 \l__enumext_compare_items_ans_int [108](#), [1417](#),
 [1423](#), [1440](#), [1449](#)
 \g__enumext_count_item_all_int [108](#), [1391](#), [1394](#),
 [1419](#), [1442](#), [2032](#), [2063](#), [3039](#)
 \g__enumext_count_item_ans_int 49, 54, 60, [108](#),
 [1401](#), [1423](#), [1449](#), [1662](#), [1929](#)
 \g__enumext_count_item_i_int [1396](#), [1443](#)
 \g__enumext_count_item_ii_int [1397](#), [1419](#), [1444](#)
 \g__enumext_count_item_iii_int [1398](#), [1420](#), [1445](#)
 \g__enumext_count_item_iv_int [1399](#), [1420](#), [1446](#)
 \g__enumext_count_item_vii_int [1400](#)
 \g__enumext_count_item_X_int [108](#)
 \g__enumext_count_level_i_int [1432](#), [1458](#)
 \g__enumext_count_level_ii_int [1433](#), [1459](#)
 \g__enumext_count_level_iii_int [1434](#), [1460](#)
 \g__enumext_count_level_iv_int [1435](#), [1461](#)
 \g__enumext_count_level_vii_int [1436](#), [1462](#),
 [3040](#)
 \g__enumext_count_level_X_int [108](#)
 \l__enumext_counter_i_tl [23](#), [275](#)
 \l__enumext_counter_ii_tl [23](#), [276](#)
 \l__enumext_counter_iii_tl [23](#), [277](#)
 \l__enumext_counter_iv_tl [23](#), [278](#)
 \l__enumext_counter_style_for_ref_vii_-
 tl [414](#), [424](#), [435](#), [437](#)
 \l__enumext_counter_style_for_ref_viii_-
 tl [441](#), [443](#)
 \l__enumext_counter_style_for_ref_X_tl [134](#)
 \c__enumext_counter_style_tl [28](#), [134](#), [381](#)
 \g__enumext_counter_styles_tl . 20, 27, [33](#), [286](#),
 [304](#)
 \l__enumext_counter_v_tl [23](#), [279](#)
 \l__enumext_counter_vi_tl [23](#), [280](#)
 \l__enumext_counter_vii_tl [23](#), [281](#), [411](#)
 \l__enumext_counter_viii_tl [23](#), [282](#), [421](#)
 \l__enumext_current_widest_dim 20, [33](#), [310](#), [455](#),
 [475](#), [491](#), [495](#)
 __enumext_default_item:n . . . [2026](#), [2026](#), [2077](#)
 __enumext_define_counters:Nn 20, [266](#), [266](#), [275](#),
 [276](#), [277](#), [278](#), [279](#), [280](#), [281](#), [282](#)
 __enumext_endminipage: . 25, [209](#), [212](#), [877](#), [2703](#),
 [3084](#)
 __enumext_fake_item: [618](#), [618](#), [2228](#)
 \l__enumext_fake_item_indent_v_dim 637, [642](#)
 \l__enumext_fake_item_indent_v_tl 639, [2085](#),
 [2089](#), [2097](#)
 \l__enumext_fake_item_indent_vii_dim 649, [654](#)
 \l__enumext_fake_item_indent_vii_tl 651, [3080](#)
 \l__enumext_fake_item_indent_viii_dim . 661,
 [666](#)
 \l__enumext_fake_item_indent_viii_tl [663](#)
 \l__enumext_fake_item_indent_X_tl [63](#)
 __enumext_fake_item_vii: [618](#), [646](#), [2264](#)
 __enumext_fake_item_viii: [618](#), [658](#), [2269](#)
 \g__enumext_footnote_arg_seq . [131](#), [1999](#), [2012](#),
 [2022](#)
 \g__enumext_footnote_int . [131](#), [2006](#), [2009](#), [2011](#),
 [2013](#)
 \g__enumext_footnote_int_seq . [131](#), [2000](#), [2013](#),
 [2018](#), [2021](#)
 __enumext_footnotes_key_bool 25
 \l__enumext_footnotes_key_bool 22, 25, [84](#), [121](#),
 [224](#), [229](#), [238](#), [3047](#), [3095](#)
 __enumext_footnotetext:nn . . . [1993](#), [1993](#), [2023](#)
 __enumext_getkeyans:nn . . . [86](#), [3146](#), [3150](#), [3150](#)
 __enumext_getkeyans_aux:n . [86](#), [3134](#), [3137](#), [3137](#)
 \l__enumext_hyperref_bool 22, 25, [121](#), [220](#), [241](#),
 [258](#), [1726](#), [1914](#), [3024](#)
 __enumext_hypertarget:nn 25, [216](#), [243](#), [247](#), [263](#)
 __enumext_if_is_int:n 191
 __enumext_if_is_int:nTF 191, [502](#), [516](#)
 \l__enumext_item_column_pos_vii_int 82, [2736](#),
 [2742](#), [2748](#), [2752](#), [2759](#), [2952](#), [3099](#), [3102](#)
 \l__enumext_item_column_pos_X_int [145](#)
 \g__enumext_item_count_all_vii_int 82, [2760](#),
 [2953](#), [3110](#), [3117](#)
 \g__enumext_item_count_all_X_int [145](#)
 __enumext_item_peek_args_vii: 82, [2954](#), [2956](#),
 [2956](#)
 __enumext_item_starred: . . 64, [2119](#), [2119](#), [2137](#)
 \l__enumext_item_starred_vii_bool 2971, [2987](#),
 [3051](#)
 \l__enumext_item_starred_X_bool [145](#)
 __enumext_item_std:w 24, 62, 63, 77, [203](#), [207](#), [2036](#),
 [2042](#), [2067](#), [2085](#), [2089](#), [2097](#), [2684](#)
 \g__enumext_item_symbol_aux_vii_tl 2995, [3053](#),
 [3056](#), [3060](#), [3062](#)
 \g__enumext_item_symbol_aux_X_tl [145](#)
 \l__enumext_item_symbol_sep_vii_dim . . 3004,
 [3012](#), [3059](#), [3061](#)
 \g__enumext_item_symbol_tl 20, 62, [28](#), [2051](#), [2125](#),
 [2142](#)
 \l__enumext_item_symbol_vii_tl [3056](#)
 \l__enumext_item_text_vii_box 3046, [3087](#), [3094](#)
 \l__enumext_item_text_X_box [145](#)
 \l__enumext_item_width_vii_dim . . 2718, [2763](#),
 [2771](#), [2772](#)
 \l__enumext_item_width_X_dim [145](#)
 \l__enumext_itemindent_X_dim [37](#)
 \l__enumext_itemsep_vii_skip [3116](#)
 \l__enumext_joined_item_aux_vii_int . . 2757,
 [2758](#), [2759](#), [2760](#), [2766](#)
 \l__enumext_joined_item_aux_X_int [145](#)
 __enumext_joined_item_vii:w . . 82, [2959](#), [2960](#),
 [2962](#), [2962](#)


```

\l__enumext_joined_item_vii_int .. 2728, 2729,
    2732, 2734, 2740, 2745, 2750, 2755, 2757, 2763
\l__enumext_joined_item_X_int ..... 145
\l__enumext_joined_width_vii_dim . 2761, 2768,
    2771, 3077, 3089
\l__enumext_joined_width_X_dim ..... 145
\__enumext_keyans_addto_prop:n 58, 1847, 1847,
    2099, 2616
\__enumext_keyans_addto_seq:n . 59, 1898, 1898,
    2101, 2618
\__enumext_keyans_anspic_code:nnn . 75, 2607,
    2610, 2610
\__enumext_keyans_check_ans:nn 60, 1932, 1932,
    2492, 2648
\__enumext_keyans_default_item:n .. 63, 2080,
    2080, 2115
\l__enumext_keyans_env_bool 13, 2350, 2503, 2587
\__enumext_keyans_fake_item: .. 618, 634, 2218
\__enumext_keyans_internal_ref: 58, 1863, 1863,
    2100, 2617
\l__enumext_keyans_level_h_int ..... 13
\l__enumext_keyans_level_int .. 13, 1152, 1649,
    1962, 2502, 2506, 2601
\__enumext_keyans_make_label: 27, 65, 2145, 2145,
    2217
\__enumext_keyans_mini_addvspace: 42, 73, 1056,
    1056, 2531
\__enumext_keyans_mini_right_cmd:n 45, 1154,
    1177, 1177
\__enumext_keyans_mini_set_vskip: 41, 994, 994,
    1058
\__enumext_keyans_multi_addvspace: . 73, 843,
    854, 2556
\__enumext_keyans_multi_set_vskip: . 38, 843,
    843, 856
\__enumext_keyans_multicols_start: 73, 2535,
    2537, 2537
\__enumext_keyans_multicols_stop: . 74, 1181,
    2562, 2562, 2586
\__enumext_keyans_parse_keys:n 2481, 2515, 2515
\l__enumext_keyans_pic_above_int . 103, 2694,
    2695, 2697
\l__enumext_keyans_pic_above_skip .. 77, 103,
    2639, 2678
\__enumext_keyans_pic_arg_two: 76, 2637, 2666,
    2666
\l__enumext_keyans_pic_below_int . 103, 2694,
    2695, 2698
\l__enumext_keyans_pic_body_seq .. 75-77, 103,
    2605, 2644, 2702
\__enumext_keyans_pic_do:n 77, 2644, 2646, 2686,
    2686, 2690
\l__enumext_keyans_pic_level_int .. 13, 1144,
    1653, 1850, 1873, 1901, 2655, 2656
\__enumext_keyans_pic_row:n 77, 2688, 2691, 2691
\__enumext_keyans_pic_safe_exec: .. 76, 2633,
    2653, 2653
\__enumext_keyans_pic_skip_abs:N .. 76, 2661,
    2661, 2677
\l__enumext_keyans_pic_width_dim . 103, 2693,
    2700
\__enumext_keyans_redefine_item: .. 64, 2103,
    2103, 2216
\__enumext_keyans_safe_exec: . 2480, 2496, 2496
\__enumext_keyans_show_left:n . 63, 1940, 1940,
    2095, 2624
\__enumext_keyans_starred_item:n .. 63, 2092,
    2092, 2111
\l__enumext_keyans_tmpa_tl .. 21, 75, 2094, 2098
\l__enumext_label_copy_i_tl .. 1780, 1876, 1880
\l__enumext_label_copy_v_tl ..... 1880
\l__enumext_label_copy_vi_tl ..... 1876
\l__enumext_label_copy_vii_tl 1755, 1766, 1797
\l__enumext_label_copy_X_tl ..... 123
\l__enumext_label_fill_left_v_tl ..... 2149
\l__enumext_label_fill_left_X_tl ..... 63
\l__enumext_label_fill_right_v_tl .... 2156
\l__enumext_label_fill_right_X_tl ..... 63
\l__enumext_label_font_style_v_tl 2150, 2628
\l__enumext_label_font_style_vii_tl ... 3065
\l__enumext_label_i_tl ..... 447
\l__enumext_label_ii_tl ..... 447
\l__enumext_label_iii_tl ..... 447
\l__enumext_label_iv_tl ..... 447
\__enumext_label_style:Nnn 20, 27, 299, 299, 314,
    452, 472, 488, 492
\l__enumext_label_v_tl .. 58, 59, 485, 1855, 1906,
    1944, 1951, 1967, 1974, 2094, 2098, 2484, 2623, 2625
\l__enumext_label_vi_tl . 58, 59, 485, 1852, 1903,
    2623, 2625, 2629
\l__enumext_label_vii_tl . 467, 2982, 3007, 3014
\l__enumext_label_viii_tl ..... 467
\l__enumext_label_width_by_box .. 33, 295, 296
\__enumext_label_width_by_box:Nn 27, 293, 293,
    298, 310, 526
\l__enumext_labelsep_i_dim ..... 1948, 1971
\l__enumext_labelsep_v_dim ..... 2546
\l__enumext_labelsep_vii_dim . 2713, 2722, 2764,
    3005, 3075, 3091
\l__enumext_labelwidth_i_dim ..... 1947, 1970
\l__enumext_labelwidth_v_dim ..... 2546
\l__enumext_labelwidth_vii_dim ... 2713, 2721,
    2764, 3068, 3072, 3090
\l__enumext_leftmargin_tmp_v_bool . 76, 2668
\l__enumext_leftmargin_tmp_X_bool ..... 37
\l__enumext_leftmargin_tmp_X_dim ..... 37
\l__enumext_leftmargin_X_dim ..... 37
\__enumext_level: 178, 178, 184, 185, 189, 390, 392,
    393, 401, 403, 621, 625, 629, 696, 700, 704, 708, 792,
    794, 796, 798, 831, 833, 835, 837, 841, 881, 884, 903,
    912, 918, 923, 927, 938, 942, 943, 948, 984, 988, 1161,
    1167, 1214, 1216, 1218, 1221, 1228, 1230, 1232, 1235,
    1409, 1410, 1412, 1557, 1565, 1569, 1573, 1818, 1821,
    1822, 2033, 2035, 2036, 2040, 2041, 2042, 2049, 2051,
    2055, 2056, 2059, 2064, 2066, 2067, 2121, 2124, 2126,
    2133, 2134, 2135, 2138, 2141, 2283, 2285, 2321, 2326,
    2327, 2328, 2330, 2334, 2339, 2340, 2341, 2343, 2354,
    2359, 2366, 2377, 2379, 2382, 2383, 2385, 2390, 2397,
    2400, 2402, 2404, 2405, 2406, 2407, 2410, 2416, 2421,
    2427, 2430, 2432, 2443, 3035, 3036
\__enumext_level_ ..... 184
\__enumext_level_end:n ..... 182, 187
\l__enumext_level_h_int 13, 409, 433, 1774, 1791,
    2304, 2352, 2847, 2848
\l__enumext_level_int 13, 180, 804, 955, 1148, 1388,
    1751, 1761, 1767, 1773, 1781, 1789, 1796, 2231, 2297,
    2298, 2313, 2357, 2412, 2470, 2510, 2597, 2856, 2933,
    2943, 3124

```

`__enumext_level_set:n` 182, 182
`__enumext_list_arg_two_i:` 2191
`__enumext_list_arg_two_ii:` 2191
`__enumext_list_arg_two_iii:` 2191
`__enumext_list_arg_two_iv:` 2191
`__enumext_list_arg_two_v:` . 64, 2191, 2486, 2669
`__enumext_list_arg_two_vii:` 2237, 2830
`__enumext_list_arg_two_viii:` 2237
`\l__enumext_listoffset_v_dim` 2548
`\l__enumext_listparindent_vii_dim` 3078
`__enumext_make_label:` 27, 62, 64, 2129, 2129, 2226
`\l__enumext_mark_answer_sym_tl` 54, 60, 98, 1471, 1627, 1833, 1955
`\l__enumext_mark_position_str` . 98, 1475, 1476, 1498, 1499, 1625
`\l__enumext_mark_ref_sym_tl` 98, 1485, 1731, 1922
`__enumext_mini_addvspace:` 41, 70, 977, 977, 2387
`__enumext_mini_addvspace_vii:` 43, 1130, 1130, 2789
`__enumext_mini_addvspace_viii:` 43, 1130, 1136
`__enumext_mini_env*` 871
`__enumext_mini_right_cmd:n` . 44, 45, 1156, 1158, 1158
`__enumext_mini_set_vskip:` ... 39, 878, 878, 979
`__enumext_mini_set_vskip_vii:` 43, 1073, 1073, 1132
`__enumext_mini_set_vskip_viii:` 43, 1073, 1095, 1138
`__enumext_minipage:w` 25, 209, 211, 873, 2700, 3077
`\l__enumext_minipage_active_v_bool` .. 73, 74, 2529, 2554, 2567, 2575
`\g__enumext_minipage_active_vii_bool` ... 79, 2800, 2805, 2817
`\l__enumext_minipage_active_vii_bool` . 2785, 2796
`\g__enumext_minipage_active_X_bool` ... 145
`\l__enumext_minipage_active_X_bool` 51
`\g__enumext_minipage_after_skip` 51, 1077, 1089, 2815
`\l__enumext_minipage_after_skip` 39, 40, 71, 74, 51, 894, 909, 929, 945, 960, 966, 972, 986, 996, 1005, 1008, 1020, 1038, 1049, 1065, 1097, 1110, 1124, 2452, 2584
`\g__enumext_minipage_center_vii_bool` . 2809, 2818
`\g__enumext_minipage_center_X_bool` ... 145
`\l__enumext_minipage_hsep_v_dim` ... 73, 2527
`\l__enumext_minipage_hsep_vii_dim` 2783
`\l__enumext_minipage_left_skip` 39, 73, 51, 886, 901, 920, 935, 982, 992, 997, 1003, 1012, 1029, 1041, 1061, 1071, 1075, 1080, 1084, 1098, 1102, 1116, 1134, 1140
`\l__enumext_minipage_left_v_dim` 73, 2525, 2533
`\l__enumext_minipage_left_vii_dim` 2779, 2791
`\l__enumext_minipage_left_X_dim` 51
`\g__enumext_minipage_right_skip` 51, 1076, 1081, 1085, 2808
`\l__enumext_minipage_right_skip` .. 39, 51, 890, 905, 925, 940, 998, 1004, 1016, 1034, 1045, 1099, 1106, 1120, 1168, 1185
`\l__enumext_minipage_right_v_dim` .. 73, 1179, 1184, 2523, 2527
`\g__enumext_minipage_right_vii_dim` 79, 2787, 2807, 2820
`\l__enumext_minipage_right_vii_dim` 79, 2777, 2782, 2788
`\g__enumext_minipage_right_X_dim` 145
`\g__enumext_minipage_right_X_skip` 145
`\g__enumext_minipage_stat_int` . 70, 73, 51, 1173, 1190, 2386, 2445, 2450, 2530, 2577, 2582
`\g__enumext_miniright_code_vii_tl` . 79, 2813, 2819
`\g__enumext_miniright_code_X_tl` 145
`__enumext_multi_addvspace:` ... 38, 70, 826, 826, 2418
`__enumext_multi_set_vskip:` .. 37, 790, 790, 828
`\l__enumext_multicols_above_ii_skip` ... 809
`\l__enumext_multicols_above_iii_skip` .. 815
`\l__enumext_multicols_above_iv_skip` ... 821
`\l__enumext_multicols_above_v_skip` 845, 859, 869
`\l__enumext_multicols_above_X_skip` 45
`\l__enumext_multicols_below_v_skip` 849, 863, 2569
`\l__enumext_multicols_below_X_skip` 45
`__enumext_multicols_start:` 70, 2392, 2394, 2394
`__enumext_multicols_stop:` 71, 1163, 2424, 2424, 2454
`__enumext_newlabel:nn` .. 22, 25, 57, 59, 251, 251, 1807, 1889
`\l__enumext_newlabel_arg_one_tl` 22, 25, 57, 59, 123, 1730, 1800, 1808, 1882, 1890, 1920
`\l__enumext_newlabel_arg_two_tl` 22, 25, 56, 58, 123, 1754, 1764, 1778, 1794, 1809, 1875, 1879, 1891
`__enumext_parse_keys:n` 2279, 2310, 2310
`__enumext_parse_keys_vii:n` .. 2825, 2862, 2862
`__enumext_parse_store_keys:n` . 68, 2316, 2319, 2319
`__enumext_parse_store_keys_vii:n` 80, 81, 2869, 2873, 2873
`\l__enumext_parsep_i_skip` . 807, 809, 958, 1006
`\l__enumext_parsep_ii_skip` 813, 815, 964
`\l__enumext_parsep_iii_skip` 819, 821, 970
`\l__enumext_parsep_vii_skip` 3079
`\l__enumext_partopsep_v_skip` .. 861, 865, 1032, 1036, 1043, 1047, 1063, 1067
`\l__enumext_partopsep_viii_skip` 1108
`__enumext_phantomsection:` 25, 216, 244, 248, 264
`__enumext_print_footnote:` ... 1993, 2016, 3097
`__enumext_print_keyans_box:NN` 54, 1619, 1619, 1632, 1820, 1946, 1969
`\l__enumext_print_keyans_i_tl` 3160, 3189
`\l__enumext_print_keyans_ii_tl` ... 3165, 3190
`\l__enumext_print_keyans_iii_tl` .. 3170, 3191
`\l__enumext_print_keyans_iv_tl` ... 3175, 3192
`\l__enumext_print_keyans_vii_tl` .. 3180, 3193
`\l__enumext_print_keyans_X_tl` 87
`__enumext_printkeyans:nnn` . 87, 3194, 3197, 3197
`__enumext_redefine_item:` . 63, 2069, 2069, 2225
`\l__enumext_ref_aux_tl` 134, 390, 392, 395, 411, 413, 416, 421, 423, 426
`\l__enumext_ref_key_arg_tl` .. 134, 384, 389, 396, 408, 417, 427
`__enumext_regex_label_ref_key:` .. 28, 29, 379, 379, 391, 412, 422
`__enumext_register_counter_style:Nn` .. 283, 283, 288, 289, 290, 291, 292

__enumext_remove_extra_parsep_vii: .. 2840, [3106](#), [3106](#)
 __enumext_renew_footnote: ... [1993](#), [1997](#), [3049](#)
 \l__enumext_resume_bool [20](#), [28](#), [1343](#), [2208](#)
 __enumext_resume_counter: . [48](#), [1309](#), [1334](#), [1334](#)
 __enumext_resume_counter_star: [1311](#)
 __enumext_resume_counter_vii: [48](#), [1318](#), [1334](#), [1345](#)
 \g__enumext_resume_int [20](#), [71](#), [28](#), [1338](#), [1349](#), [2209](#), [2458](#)
 \l__enumext_resume_vii_bool ... [28](#), [1354](#), [2255](#)
 \g__enumext_resume_vii_int .. [81](#), [28](#), [2256](#), [2914](#)
 __enumext_safe_exec: [2278](#), [2295](#), [2295](#)
 __enumext_safe_exec_vii: ... [2824](#), [2845](#), [2845](#)
 __enumext_set_error:nn [3274](#), [3284](#), [3286](#)
 __enumext_set_label_ref:n ... [28](#), [387](#), [387](#), [459](#)
 __enumext_set_label_ref_h:n . [29](#), [406](#), [406](#), [479](#)
 __enumext_set_parse:n [3257](#), [3274](#), [3274](#)
 \l__enumext_setkey_tmpa_int ... [82](#), [3250](#), [3254](#)
 \l__enumext_setkey_tmpa_seq [82](#), [3248](#), [3258](#), [3264](#), [3266](#), [3268](#), [3281](#)
 \l__enumext_setkey_tmpa_tl [82](#), [3256](#), [3260](#)
 \l__enumext_setkey_tmpb_seq [82](#), [3249](#), [3252](#), [3256](#), [3257](#)
 \l__enumext_setkey_tmpb_tl [82](#), [3276](#), [3278](#), [3279](#)
 \l__enumext_show_answer_bool .. [98](#), [1479](#), [1483](#), [1502](#), [1506](#), [1827](#), [1942](#), [2620](#)
 __enumext_show_length:nnn .. [34](#), [197](#), [197](#), [3314](#), [3315](#), [3316](#), [3317](#), [3318](#), [3319](#), [3320](#), [3321](#), [3322](#), [3323](#), [3329](#), [3330](#), [3331](#), [3332](#), [3333](#), [3334](#), [3335](#), [3336](#), [3337](#), [3338](#)
 \l__enumext_show_position_bool [98](#), [1480](#), [1482](#), [1503](#), [1505](#), [1831](#), [1953](#), [2621](#)
 \g__enumext_standar_bool [13](#), [2307](#)
 \l__enumext_standar_bool . [13](#), [1759](#), [1772](#), [1788](#), [2300](#), [2457](#), [2855](#)
 __enumext_standard_item_vii:w .. [82](#), [83](#), [2967](#), [2969](#), [2969](#)
 \g__enumext_starred_bool . [80](#), [81](#), [13](#), [1387](#), [1750](#), [1760](#), [1790](#), [2435](#), [2859](#), [2921](#), [2926](#)
 \l__enumext_starred_bool . [80](#), [81](#), [13](#), [1683](#), [1691](#), [1775](#), [1816](#), [2303](#), [2852](#), [2927](#)
 __enumext_starred_columns_set_vii: .. [2707](#), [2707](#), [2833](#)
 __enumext_starred_item:nn ... [2045](#), [2045](#), [2075](#)
 __enumext_starred_item_vii:w [82](#), [83](#), [2966](#), [2985](#), [2985](#)
 __enumext_starred_item_vii_aux_i:w .. [2985](#), [2990](#), [2993](#)
 __enumext_starred_item_vii_aux_ii:w . [2985](#), [2991](#), [2996](#), [2998](#)
 __enumext_starred_item_vii_aux_iii:w [2985](#), [3001](#), [3010](#)
 __enumext_starred_joined_item_vii:n . [78](#), [82](#), [2726](#), [2726](#), [2964](#)
 __enumext_start_from:NNn [31](#), [500](#), [500](#), [513](#), [535](#)
 __enumext_start_item_tmp_vii: [80](#), [2836](#), [2949](#), [2949](#)
 __enumext_start_item_vii:w . [83](#), [84](#), [2977](#), [2982](#), [3007](#), [3014](#), [3016](#), [3016](#)
 __enumext_start_list:nn [24](#), [66](#), [76](#), [203](#), [205](#), [2282](#), [2483](#), [2634](#), [2828](#)
 __enumext_start_mini_vii: . [81](#), [2775](#), [2775](#), [2906](#)
 __enumext_start_store_level: . [69](#), [2281](#), [2348](#), [2348](#)
 __enumext_start_store_level_vii: . [82](#), [2827](#), [2929](#), [2929](#)
 \l__enumext_start_X_int [63](#), [530](#)
 __enumext_stop_item_tmp_vii: . [80](#), [82](#), [84](#), [2835](#), [2839](#), [2951](#), [3018](#)
 __enumext_stop_item_vii: [84](#), [85](#), [3018](#), [3082](#), [3082](#)
 __enumext_stop_list: .. [24](#), [203](#), [206](#), [2291](#), [2493](#), [2647](#), [2841](#)
 __enumext_stop_mini_vii: [79](#), [81](#), [2794](#), [2794](#), [2911](#)
 __enumext_stop_store_level: .. [69](#), [2292](#), [2348](#), [2364](#)
 __enumext_stop_store_level_vii: .. [82](#), [2842](#), [2929](#), [2939](#)
 \l__enumext_store_active_bool [21](#), [47](#), [68](#), [80](#), [75](#), [1324](#), [1336](#), [1347](#), [1645](#), [2314](#), [2350](#), [2498](#), [2505](#), [2593](#), [2651](#), [2867](#), [2931](#), [2941](#)
 __enumext_store_addto_prop:n [52](#), [58](#), [1536](#), [1536](#), [1545](#), [1670](#), [1861](#)
 __enumext_store_addto_seq:n [52](#), [59](#), [1546](#), [1546](#), [1552](#), [1559](#), [1573](#), [1581](#), [1590](#), [1608](#), [1616](#), [1734](#), [1925](#)
 \l__enumext_store_ans_bool ... [1362](#), [1407](#), [1555](#), [1579](#), [1586](#), [1614](#), [1658](#), [3033](#)
 \l__enumext_store_anskey_arg_tl .. [21](#), [55](#), [75](#), [1676](#), [1685](#), [1687](#), [1693](#), [1701](#), [1704](#), [1714](#), [1719](#), [1722](#), [1728](#), [1734](#)
 __enumext_store_anskey_code:nnnn [54](#), [55](#), [1664](#), [1668](#), [1668](#)
 __enumext_store_anskey_show_left:n [57](#), [1675](#), [1825](#), [1825](#)
 __enumext_store_anskey_show_wrap:n [57](#), [1813](#), [1813](#), [1829](#), [1844](#)
 \l__enumext_store_columns_break_bool . [1639](#), [1682](#)
 \l__enumext_store_columns_join_int [75](#), [1690](#), [1695](#)
 \l__enumext_store_columns_sep_vii_bool [2888](#)
 \l__enumext_store_columns_sep_vii_dim [2893](#), [2897](#)
 \l__enumext_store_columns_sep_X_bool ... [87](#)
 \l__enumext_store_columns_sep_X_dim [87](#)
 \l__enumext_store_columns_vii_bool ... [2875](#)
 \l__enumext_store_columns_vii_int [2880](#), [2884](#)
 \l__enumext_store_columns_X_bool [87](#)
 \l__enumext_store_columns_X_int [87](#)
 __enumext_store_internal_ref: [56](#), [1673](#), [1736](#), [1736](#)
 \l__enumext_store_item_symbol_sep_dim [1637](#), [1711](#), [1716](#)
 \l__enumext_store_item_symbol_tl . [1635](#), [1702](#), [1706](#)
 \l__enumext_store_keyans_label_tl [21](#), [58–60](#), [75](#), [1849](#), [1852](#), [1855](#), [1859](#), [1861](#), [1900](#), [1903](#), [1906](#), [1910](#), [1916](#), [1925](#), [1926](#)
 __enumext_store_level_close: . [52](#), [1553](#), [1577](#), [2368](#)
 __enumext_store_level_close_vii: [1584](#), [1612](#), [2945](#)
 __enumext_store_level_open: .. [51](#), [52](#), [68](#), [1553](#), [1553](#), [2355](#), [2360](#)
 __enumext_store_level_open_vii: .. [81](#), [1584](#), [1584](#), [2935](#)
 \g__enumext_store_name_tl [21](#), [71](#), [75](#), [1426](#), [1430](#), [1452](#), [1456](#), [2438](#), [2476](#), [2924](#), [3130](#)

<code>\l__enumext_store_name_tl</code>	21, 47, <u>75</u> , 1323, 1340, 1351, 1538, 1539, 1540, 1542, 1548, 1549, 1550, 1802, 1803, 1839, 1884, 1885, 1961, 2438, 2459, 2462, 2915, 2918, 2924
<code>\l__enumext_store_opt_vii_tl</code>	. 1588, 1598, 1604, 1608, 2882, 2895
<code>\l__enumext_store_opt_X_tl</code> <u>87</u>
<code>\l__enumext_store_ref_key_bool</code>	55, 1488, 1671, 1725, 1893, 1913
<code>\l__enumext_store_upper_level_X_bool</code>	... <u>87</u>
<code>\l__enumext_store_write_aux_file_tl</code>	22, 57, 59, <u>123</u> , 1805, 1811, 1887, 1895
<code>__enumext_storing_set:n</code>	47, 48, 1307, 1316, <u>1321</u> , 1321
<code>\l__enumext_the_counter_vii_tl</code> 413
<code>\l__enumext_the_counter_viii_tl</code> 423
<code>\l__enumext_the_counter_X_tl</code> <u>134</u>
<code>__enumext_tmp:n</code>	23, 27, 37, 44, 45, 50, 57, 62, 63, 74, 87, 97, 115, 120, 126, 130, 138, 144, 145, 164, 613, 617, 1356, 1370, 1464, 1492, 1493, 1510, 1738, 1745, 1746, 1767, 1781, 1784, 1796, 1865, 1872, 2191, 2236, 2237, 2275
<code>__enumext_tmp:nn</code>	315, 336, 337, 365, 366, 378, 530, 549, 594, 612, 670, 678, 679, 693, 758, 774, 775, 789, 1194, 1210, 1511, 1535, 1977, 1992
<code>__enumext_tmp:nnn</code>	447, 463, 464, 465, 466, 467, 483, 484
<code>__enumext_tmp:nnnnn</code>	550, 575, 578, 581, 583, 585, 588, 591
<code>__enumext_tmp:w</code> 3143, 3146
<code>\l__enumext_tmpa_vii_int</code> 2717, 2720
<code>\l__enumext_tmpa_X_int</code> <u>145</u>
<code>\l__enumext_topsep_v_skip</code>	847, 851, 1001, 1014, 1022, 1027, 1047, 1051, 2650, 2681
<code>\l__enumext_topsep_vii_skip</code>	.. 1078, 1087, 1091
<code>\l__enumext_topsep_viii_skip</code>	. 1100, 1122, 1126
<code>__enumext_use_key_ref:</code>	... 29, <u>399</u> , 399, 2227
<code>__enumext_use_key_ref_h:</code>	.. 29, <u>431</u> , 431, 2261
<code>\l__enumext_vspace_a_star_v_bool</code> 1243
<code>\l__enumext_vspace_a_star_vii_bool</code>	... 1265
<code>\l__enumext_vspace_a_star_viii_bool</code>	... 1276
<code>\l__enumext_vspace_a_star_X_bool</code> <u>63</u>
<code>__enumext_vspace_above:</code>	.. 45, <u>1211</u> , 1211, 2373
<code>__enumext_vspace_above_v:</code>	. 46, <u>1239</u> , 1239, 2521
<code>\l__enumext_vspace_above_v_skip</code>	.. 1241, 1245, 1247
<code>__enumext_vspace_above_vii:</code>	.. 46, <u>1261</u> , 1261, 2904
<code>\l__enumext_vspace_above_vii_skip</code>	1263, 1267, 1269
<code>__enumext_vspace_above_viii:</code>	. 46, <u>1261</u> , 1272
<code>\l__enumext_vspace_above_viii_skip</code>	1274, 1278, 1280
<code>\l__enumext_vspace_b_star_v_bool</code> 1254
<code>\l__enumext_vspace_b_star_vii_bool</code>	... 1287
<code>\l__enumext_vspace_b_star_viii_bool</code>	... 1298
<code>\l__enumext_vspace_b_star_X_bool</code> <u>63</u>
<code>__enumext_vspace_below:</code>	.. 46, <u>1225</u> , 1225, 2456
<code>__enumext_vspace_below_v:</code>	. 46, <u>1250</u> , 1250, 2589
<code>\l__enumext_vspace_below_v_skip</code>	.. 1252, 1256, 1258
<code>__enumext_vspace_below_vii:</code>	.. 47, <u>1283</u> , 1283, 2913
<code>\l__enumext_vspace_below_vii_skip</code>	1285, 1289, 1291
<code>__enumext_vspace_below_viii:</code>	. 47, <u>1283</u> , 1294
<code>\l__enumext_vspace_below_viii_skip</code>	1296, 1300, 1302
<code>__enumext_widest_from:nnn</code>	.. 31, <u>514</u> , 514, 529, 541
<code>\g__enumext_widest_label_tl</code>	20, 27, <u>33</u> , 303, 307, 311
<code>\l__enumext_wrap_label_opt_v_bool</code>	... 2088
<code>\l__enumext_wrap_label_opt_vii_bool</code>	83, 2976
<code>\l__enumext_wrap_label_opt_X_bool</code> <u>63</u>
<code>\l__enumext_wrap_label_v_bool</code>	2084, 2088, 2096, 2151
<code>\l__enumext_wrap_label_vii_bool</code>	83, 2975, 2980, 2988, 3066
<code>\l__enumext_wrap_label_X_bool</code> <u>63</u>
<code>__enumext_wrapper_label_v:n</code>	... 2153, 2629
<code>__enumext_wrapper_label_vii:n</code>	... 3069
<code>__enumext_zero_parsep:</code>	... 40, 898, <u>953</u> , 953
<code>enumext*</code> 4, <u>2822</u>
<code>enumXi</code> <u>275</u>
<code>enumXii</code> <u>275</u>
<code>enumXiii</code> <u>275</u>
<code>enumXiv</code> <u>275</u>
<code>enumXv</code> <u>275</u>
<code>enumXvi</code> <u>275</u>
<code>enumXvii</code> <u>275</u>
<code>enumXviii</code> <u>275</u>
Environments provide by enumext :	
<code>enumext*</code>	19–21, 23, 24, 26, 28–30, 33, 34, 36, 43, 46–49, 51–56, 61, 69, 80–82, 84–86, 89, 91
<code>enumext</code>	19–21, 23, 24, 26, 27, 29, 30, 32–42, 44–49, 51–56, 61–66, 68, 69, 71, 72, 76, 77, 79, 82, 86, 89, 90
<code>keyans*</code>	19–21, 23, 24, 26, 28–30, 33, 34, 36, 43, 46–48, 51, 52, 61, 89, 91
<code>keyanspic</code>	19–22, 26, 27, 30, 44, 47, 48, 52, 58–60, 74–77, 90
<code>keyans</code>	19–22, 24, 26, 27, 30, 32–36, 38, 41, 42, 44–48, 51, 52, 58–60, 64–66, 72, 74–76, 79, 89, 90
Environments:	
<code>enumext*</code> 67
<code>keyans*</code> 67
<code>list</code> 23, 24, 65, 66, 68
<code>lrbox</code> 77, 84, 85
<code>minipage</code> 23–25, 36, 39, 74, 76, 77, 84, 85
<code>multicols</code> 37–39, 44, 70, 71, 73, 74
exp commands:	
<code>\exp_after:wN</code> 3146
<code>\exp_args:Ne</code> 2312, 3134
<code>\exp_not:N</code>	142, 306, 395, 416, 426, 627, 641, 642, 653, 654, 665, 666, 1730, 1836, 1837, 1918, 1958, 1959, 3143
<code>\exp_not:n</code>	395, 396, 416, 417, 426, 427, 628, 1519, 1526, 1695, 1706, 1716, 1730, 1731, 1808, 1890, 1920, 1922, 2330, 2343, 2884, 2897
F	
<code>\fbox</code> 1469
file commands:	
<code>\file_input_stop:</code> 3431
<code>first</code> <u>679</u>
<code>font</code> <u>315</u>
<code>\footnote</code> 61
<code>\footnote</code> 61, 2001

\footnotemark 2011
 \footnotesize 1837, 1959
 \footnotetext 1995

G

\getkeyans 12, 86, 3132
 group commands:
 \group_begin: 1657, 1835, 1957, 3045, 3064, 3154, 3188
 \group_end: 1666, 1842, 1965, 3074, 3086, 3156, 3195

H

\hbadness 3093
 hbox commands:
 \hbox_set:Nn 295
 \hfill 345, 349, 354, 355, 1165, 1183, 1730, 1918, 2799
 hook commands:
 \hook_gput_code:nnn 172, 176, 214
 \hook_gset_rule:nnnn 215
 \hspace 3104
 \hyperlink 56, 59
 \hyperlink 1730, 1918
 \hypertarget 25
 \hypertarget 243

I

\IfHyperBoolean 221
 \IfPackageLoadedTF 4, 218, 231
 \ignorespaces 630
 int commands:
 \int_add:Nn 2759
 \int_case:nn 804, 955
 \int_compare:nNnTF . 409, 433, 880, 999, 1144, 1148,
 1152, 1422, 1448, 1649, 1653, 1850, 1873, 1901, 2298,
 2352, 2357, 2396, 2412, 2426, 2445, 2470, 2506, 2510,
 2539, 2564, 2577, 2597, 2601, 2656, 2729, 2739, 2755,
 2848, 2933, 2943, 3099, 3108, 3124, 3254
 \int_compare_p:nNn . 1388, 1751, 1761, 1773, 1774,
 1789, 1791, 2304, 2856
 \int_decr:N 2758
 \int_eval:n 1542, 1803, 1837, 1885, 1959, 2209, 2212,
 2256, 2259, 2747
 \int_from_alph:n 508, 522
 \int_from_roman:n 510, 524
 \int_gadd:Nn 1409, 2760, 3035
 \int_gincr:N 1412, 1662, 1929, 2032, 2033, 2063, 2064,
 2386, 2530, 2953, 3039, 3040
 \int_gset:Nn 1338, 1349, 2009
 \int_gset_eq:NN 1391, 1394, 1396, 1397, 1398, 1399,
 1400, 1401, 2006, 2458, 2461, 2914, 2917
 \int_gzero:N 1173, 1190, 1432, 1433, 1434, 1435, 1436,
 1458, 1459, 1460, 1461, 1462, 2450, 2582, 3117
 \int_if_exist:NnTF 1325, 1373, 1375, 1377, 1379, 1381,
 1383, 2459, 2915
 \int_incr:N 2297, 2502, 2655, 2847, 2952
 \int_mod:nn 3110
 \int_new:N 13, 14, 15, 16, 17, 29, 31, 51, 67, 79, 84, 92,
 105, 106, 112, 113, 114, 117, 118, 131, 148, 149, 150,
 151, 152, 1327, 1374, 1376, 1378, 1380, 1382, 1384
 \int_set:Nn 504, 508, 510, 1417, 1440, 1516, 1690, 2694,
 2695, 2717, 2728, 2734, 2750, 3093, 3250
 \int_set_eq:NN 2325, 2757, 2879
 \int_step_function:nnN 1767, 1781, 1796
 \int_step_inline:nnn 2696, 3277
 \int_to_roman:n 180, 1747, 1785
 \int_use:N . 881, 1340, 1351, 1410, 2212, 2231, 2259,
 2313, 2397, 2406, 2421, 2427, 2732, 2733, 2745, 3036

\int_zero:N 3102
 \c_one_int 2717, 2736, 2742, 2748, 2752, 2755
 \c_zero_int 1388, 1751, 1761, 1773, 1774, 1789, 1791,
 2304, 2856, 2933, 2943, 3113
 \item 24, 35, 36, 53, 62, 74, 75, 77, 80
 \item . 62, 63, 82, 84, 207, 1561, 1567, 1592, 1600, 1687, 1903,
 1906, 2071, 2105, 2834, 2836
 \item* 5, 10, 2103
 item-pos* 1977
 item-sym* 1977
 \itemindent 20, 66
 \itemindent 65
 itemindent 594
 \itemsep 76, 77
 \itemsep 2670, 2676
 \itemwidth 2724, 2768, 2772

K

keyans 10, 2478
 keyans* 10
 keyanspic 11, 2631

Keys for environments provide by [enumext](#):

above* 21, 45, 46
 above 21, 45, 46, 69, 73, 81
 after 34–36, 71, 74, 81
 align 21, 28, 64, 84
 before* 34, 35, 69, 81
 before 34–36, 73
 below* 21, 45–47
 below 21, 45–47, 71, 74, 81
 check-ans . 21–23, 47, 49, 54, 60, 62, 63, 69, 71, 85, 89
 columns-sep* 21, 51, 68, 81
 columns-sep 36, 52, 68, 70, 73, 81
 columns* 21, 51, 68, 81
 columns 20, 36, 39, 45, 52, 68, 70, 73, 81
 first 34–36, 84
 font 27, 64, 84
 item-pos* 54, 55, 61
 item-sym* 20, 54, 55, 61, 62
 item*-sep 62
 itemindent 21, 33, 64, 84
 itemsep 32, 67
 labelsep 27, 62, 66, 84
 labelwidth 26, 27, 30, 31, 66
 label 20, 26, 27, 30, 31, 77
 lisparindent 67
 list-indent 20, 33, 76
 list-offset 33
 listparindent 33, 84
 mark-ans 22, 51, 57
 mark-pos 51
 mark-ref 22, 51, 56, 58
 mini-env 20, 36, 39, 44, 45, 61, 69, 73, 79, 81
 mini-sep 20, 36, 69, 73
 miniright* 20, 36
 miniright 20, 36, 43, 79
 minirigth* 23
 minirigth 23
 no-store 22, 48, 49
 noitemsep 32, 40
 nosep 32, 40
 parindent 67
 parsep 32, 67, 84
 partopsep 32
 ref 23, 28, 29

resume	20, 47, 48, 66, 71, 81
rightmargin	33
save-ans	21, 47, 48, 52, 54, 58, 59, 63, 71, 72, 75, 81, 86
save-key	22
show-ans	22, 51, 54, 55, 57, 63
show-length	24, 34, 66, 89
show-pos	22, 51, 54, 55, 57, 63
start	21, 24, 31, 32, 66
store-brk	54, 55
store-ref	22, 25, 51, 55, 56, 58, 59, 63
topsep	32
widest	20, 24, 31, 32
wrap-ans	51, 54, 57
wrap-label*	27, 62, 64, 83, 84
wrap-label	27, 64, 83, 84
keys commands:	
\keys_define:nn	317, 339, 368, 449, 469, 485, 532, 552, 596, 615, 672, 681, 760, 777, 1196, 1305, 1314, 1358, 1466, 1495, 1513, 1633, 1979, 3158, 3220
\l_keys_key_str	3299
\keys_set:nn	331, 784, 1201, 1206, 1679, 2312, 2517, 2866, 3222, 3223, 3224, 3225, 3226, 3227, 3228, 3229, 3230, 3231, 3232, 3233, 3271
L	
l internal commands:	
\l_enumext_store_internal_ref:	55
label	447, 467, 485
Labels provide by enumext:	
\Alph*	26, 27
\Roman*	26, 27
\alph*	26, 27
\arabic*	26-28
\roman*	26, 27
\labelsep	77
\labelsep	2671, 2674
labelsep	315
\labelwidth	27, 77
\labelwidth	2671, 2672
labelwidth	315
\leftmargin	20, 66
\leftmargin	65, 2671
legacy commands:	
\legacy_if:nTF	3019, 3022
\legacy_if_gset_false:n	874
\legacy_if_set_false:n	3021
\legacy_if_set_true:n	2981, 3006, 3013, 3026
\linewidth	69, 73
\linewidth	2381, 2527, 2693, 2720, 2781
\list	24
\list	205
list-indent	594
list-offset	594
\listparindent	2673
listparindent	594
\lrbox	3046
M	
\makebox	77
\makebox	1623, 1625, 2125, 3060, 3068, 3072
\makelabel	62, 64, 65, 77
\makelabel	64, 65, 2131, 2147
\makesavenoteenv	237
mark-ans	1464
mark-pos	1464, 1493

mark-ref	1464
mini-env	758
mini-sep	758
\minipage	25
\minipage	211
\miniright	8, 44, 1142, 2448, 2580
\miniright*	8
mode commands:	
\mode_if_vertical:TF	829, 857, 980, 1059
\mode_leave_vertical:	627, 641, 653, 665, 1592, 1600, 1621, 2123, 3058
msg commands:	
\msg_error:nn	2508, 2512, 2599, 2658, 2850, 3234
\msg_error:nnn	1146, 1150, 1175, 1192, 3148, 3153, 3217, 3287
\msg_error:nnnn	1647, 1651, 1655, 2500, 2595, 2603
\msg_fatal:nn	2299
\msg_fatal:nnn	269
\msg_line_context:	3303, 3308, 3313, 3328, 3351, 3355, 3359, 3364, 3369, 3374, 3379, 3383, 3388, 3393, 3397, 3402, 3406, 3411, 3416, 3421, 3425, 3429
\msg_new:nnn	3288, 3293, 3297, 3301, 3306, 3311, 3326, 3341, 3348, 3353, 3357, 3362, 3367, 3372, 3377, 3381, 3386, 3391, 3395, 3400, 3404, 3409, 3414, 3419, 3423, 3427
\msg_term:nnn	1425, 1451
\msg_term:nnnn	2221, 2231, 2266, 2271
\msg_warning:nn	2447, 2579
\msg_warning:nnn	1429, 1455
\msg_warning:nnnn	1936, 2163, 2168, 2731, 2744
\multicolsep	70, 73
\multicolsep	2411, 2552
N	
\NeedsTeXFormat	3
\newcounter	272
\NewDocumentCommand	1142, 1643, 2591, 3132, 3186, 3241
\NewDocumentEnvironment	2276, 2478, 2631, 2822
\newlabel	26
\newlabel	255
no-store	1356
\noindent	80
\noindent	2388, 2532, 2790, 2835, 3101
\nointerlineskip	2388, 2532, 2790
noitemsep	550
\nopagebreak	840, 868, 991, 1070, 1133, 1139
\normalfont	1836, 1958
nosep	550
P	
Packages:	
enumext	19, 47, 65, 75, 89
enumitem	26
expl3	77
footnotehyper	25
hyperref	22, 23, 25, 26, 29, 56, 59, 84
lua-visual-debug	39
multicol	19
shortlst	77
\par	840, 868, 991, 1070, 1133, 1139, 1168, 1185, 1815, 2432, 2452, 2569, 2584, 2705, 2808, 2815, 3101, 3115
\parindent	3078
\parsep	37, 40, 76, 77
\parsep	1593, 1601, 2251, 2670, 2677, 2682
parsep	550

\parskip 3079

\partopsep 77

\partopsep 2252, 2675

partopsep 550

peek commands:

 \peek_meaning:NTF 2958, 2972, 2989, 3000

 \peek_meaning_remove:NTF 2965

 \peek_remove_spaces:n 2109

\phantomsection 25

\phantomsection 244

prg commands:

 \prg_do_nothing: 248

 \prg_new_protected_conditional:Npnn ... 191

 \prg_replicate:nn 200, 3346

 \prg_return_false: 195

 \prg_return_true: 194

\printkeyans 12, 86, 3186

prop commands:

 \prop_count:N 1542, 1803, 1839, 1885, 1961

 \prop_gput:Nnn 1540

 \prop_if_exist:NTF 1538, 3152

 \prop_item:Nn 3155

 \prop_new:N 1539

\ProvidesExplPackage 8

R

\raggedcolumns 2420, 2558

\ref 56, 58

ref 447, 467

\refstepcounter 3028

regex commands:

 \regex_match:nnTF 193, 507, 509, 521, 523, 2323, 2336, 2877, 2890

 \regex_replace_once:nnN 383

\renewcommand 395, 416, 426

\RenewDocumentCommand ... 2001, 2071, 2105, 2131, 2147

\RequirePackage 6

resume 1305

resume* 1305

rightmargin 594

\Roman 27, 31

\Roman 291

\roman 27, 31

\roman 292, 465, 3174

S

save-ans 1305

scan commands:

 \scan_stop: 77, 2684, 2834, 3143, 3146

seq commands:

 \seq_clear:N 3248

 \seq_const_from_clist:Nn 3236

 \seq_count:N 2644, 3252

 \seq_gclear:N 1999, 2000

 \seq_gput_right:Nn 1550, 2012, 2013

 \seq_if_empty:NTF 2018, 3201, 3266

 \seq_if_exist:NTF 1548, 3199

 \seq_item:Nn 2702

 \seq_map_function:NN 3257

 \seq_map_inline:Nn .. 3206, 3211, 3245, 3267, 3268

 \seq_map_pairwise_function:NNN 2020

 \seq_new:N 85, 86, 103, 132, 133, 1549

 \seq_pop_left:NN 3256

 \seq_put_right:Nn 2605, 3264, 3281

 \seq_set_from_clist:Nn 3249

 \seq_set_map_e:NNn 3258

\setcounter .. 518, 522, 524, 2209, 2211, 2256, 2258, 2649

\setenumext .. 5-7, 87, 3162, 3167, 3172, 3177, 3182, 3241

\setlength 1594, 1602

show-ans 1464, 1493

show-length 670

skip commands:

 \skip_add:Nn . 809, 815, 821, 831, 835, 859, 863, 960, 966, 972, 982, 986, 1008, 1061, 1065, 2670

 \skip_eval:n 1593, 1601

 \skip_gset:Nn 1081, 1085, 1089

 \skip_gzero_new:N 1076, 1077

 \skip_horizontal:N 642, 654, 666, 3061, 3075

 \skip_horizontal:n 628, 1622, 1630, 2124, 2126, 3059

 \skip_if_eq:nnTF . 807, 813, 819, 883, 917, 958, 964, 970, 1001, 1006, 1027, 1078, 1100, 1213, 1227, 1241, 1252, 1263, 1274, 1285, 1296

 \skip_new:N 47, 48, 52, 53, 54, 55, 56, 107, 162

 \skip_set:Nn . 792, 796, 845, 849, 886, 890, 894, 901, 905, 909, 920, 925, 929, 935, 940, 945, 1003, 1004, 1005, 1012, 1016, 1020, 1029, 1034, 1038, 1041, 1045, 1049, 1080, 1084, 1102, 1106, 1110, 1116, 1120, 1124, 2664, 2678

 \skip_set_eq:NN 2204, 2250, 2251, 3078, 3079

 \skip_use:N 794, 798, 833, 837, 841, 861, 865, 884, 903, 912, 918, 923, 927, 938, 942, 943, 948, 984, 988, 1014, 1214, 1218, 1221, 1228, 1232, 1235, 2432

 \skip_zero:N 2252, 2411, 2552, 2675, 2676

 \skip_zero_new:N . 996, 997, 998, 1075, 1097, 1098, 1099

 \c_zero_skip . 807, 813, 819, 884, 918, 958, 964, 970, 1001, 1006, 1027, 1078, 1100, 1214, 1228, 1241, 1252, 1263, 1274, 1285, 1296

\small 3164, 3169, 3174, 3179, 3184

\star 1983

start 530

\stepcounter 2005, 2612

store-ref 1464

str commands:

 \c_backslash_str 3355, 3364, 3365, 3369, 3370, 3374, 3375, 3406, 3407, 3411, 3416, 3417

 \c_colon_str 1802, 1884, 3143

 \str_count:n 200, 3346

 \str_if_eq:nnTF 2214, 2262

 \str_if_eq_p:nn 2207, 2255

 \str_if_in:nnTF 3139

 \str_new:N 102, 157

 \str_set:Nn ... 371, 372, 373, 1475, 1476, 1498, 1499

\string 237

\strutbox 888, 892, 896, 907, 911, 922, 931, 937, 947, 960, 966, 972, 1003, 1004, 1005, 1008, 1018, 1022, 1031, 1038, 1043, 1051, 1080, 1081, 1084, 1091, 1104, 1112, 1118, 1126, 2680

T

TeX and L^AT_EX commands:

 \@auxout 253

 \protected@write 253

text commands:

 \text_expand:n 3135

\textasteriskcentered 1472, 1486

\thepage 259

tl commands:

 \c_space_tl 1859, 1910, 1951, 1974, 3313, 3328

 \tl_clear:N 344, 350, 1676, 1849, 1900

<code>\tl_clear_new:N</code>	301
<code>\tl_const:Nn</code>	134, 285
<code>\tl_gclear:N</code> ...	1938, 2142, 2476, 2819, 3062, 3130
<code>\tl_gput_right:Nn</code>	286
<code>\tl_greplace_all:Nnn</code>	307
<code>\tl_gset:Nn</code>	1926, 2438, 2924, 2995
<code>\tl_gset_eq:NN</code>	303, 2051, 3055
<code>\tl_if_blank:nTF</code>	3053
<code>\tl_if_empty:nTF</code> .	401, 435, 441, 1557, 1588, 1702, 1934, 2121, 3279
<code>\tl_if_novalue:nTF</code> ..	1677, 1688, 1857, 1908, 1950, 1973, 2003, 2028, 2047, 2052, 2082, 2642, 2864, 3243
<code>\tl_map_inline:Nn</code>	304, 381
<code>\tl_new:N</code> 25, 32, 34, 35, 68, 69, 70, 76, 77, 78, 80, 81, 82, 83, 89, 90, 100, 101, 111, 123, 124, 125, 128, 136, 137, 140, 141, 156, 159	
<code>\tl_put_left:Nn</code>	1565, 1598, 1685, 1944, 1967
<code>\tl_put_right:Nn</code> 302, 393, 414, 424, 1517, 1524, 1569, 1604, 1687, 1693, 1701, 1704, 1714, 1719, 1722, 1728, 1754, 1764, 1778, 1794, 1800, 1805, 1852, 1855, 1859, 1875, 1879, 1882, 1887, 1903, 1906, 1910, 1916, 1951, 1974, 2328, 2341, 2882, 2895, 3160, 3165, 3170, 3175, 3180	
<code>\tl_remove_all:Nn</code>	3278
<code>\tl_remove_once:Nn</code>	1742, 1869
<code>\tl_replace_all:Nnn</code>	306
<code>\tl_reverse:N</code>	1741, 1743, 1868, 1870
<code>\tl_set:Nn</code> 142, 271, 345, 349, 354, 355, 389, 408, 625, 639, 651, 663, 1323, 1471, 1485, 1833, 1955, 2049, 3276	
<code>\tl_set_eq:NN</code> 312, 390, 392, 411, 413, 421, 423, 1740, 1867, 2094, 2098, 2623, 2625	
<code>\tl_to_str:n</code>	3135
<code>\tl_trim_spaces:n</code>	302, 3264, 3276, 3282
<code>\tl_use:N</code> .	308, 311, 403, 437, 443, 696, 700, 704, 708, 712, 716, 720, 724, 728, 732, 736, 740, 744, 748, 752, 756, 1627, 1747, 1755, 1766, 1780, 1785, 1797, 2036, 2042, 2067, 2085, 2089, 2097, 2125, 2133, 2134, 2141, 2149, 2150, 2156, 2283, 2484, 2628, 2813, 3065, 3076, 3080, 3189, 3190, 3191, 3192, 3193, 3260
token commands:	
<code>\token_to_str:N</code>	255
<code>\topsep</code>	1594, 1602
<code>topsep</code>	<u>550</u>
<code>\typeout</code>	223, 226, 233, 236, 237
U	
<code>\u</code>	384
use commands:	
<code>\use:N</code>	201, 2138, 2285
<code>\use:n</code>	3141
<code>\use_none:nn</code>	247
<code>\usecounter</code>	2205, 2253
V	
<code>\value</code>	2458, 2463, 2914, 2919
<code>\vspace</code> 875, 1218, 1221, 1232, 1235, 1245, 1247, 1256, 1258, 1267, 1269, 1278, 1280, 1289, 1291, 1300, 1302, 1593, 1601, 2639, 2650, 3116	
W	
<code>widest</code>	<u>530</u>
<code>wrap-ans</code>	<u>1464</u>
<code>wrap-label</code>	<u>315</u>
<code>wrap-label*</code>	<u>315</u>