The docindex package

Lars Hellström

8 July 2003

Abstract

The docindex package implements template-based formatting of indices and lists of changes/glossaries. In addition to this, the control structures employed also provide for a couple of new features, such as automatic collapsing of trivial index levels.

Contents

1 Introduction **1**

2 Usage **3**
2.1 Straightforward usage 3
2.2 Multiple indices 3
2.3 Configuration 4

3 Implementation **7**
3.1 docstrip modules 7
3.2 Initial stuff 7
3.3 Index style files 8
3.4 Template mechanisms 12
3.5 Templates for index item formatting 14
  3.5.1 The `indexitem` template type 14
  3.5.2 The `docindex` template type 22

4 Notes and acknowledgements **28**

1 Introduction

In automatically generated indices with multi-level entries, such as the list of changes of a doc document, it often happens that some entries are uniquely identified by their primary level sort keys, although there are sort keys and text for additional levels. If then the formatting is designed for entries that are uniquely identified only when their secondary or tertiary sort keys are considered, one ends up with a couple of entries that look rather peculiar (building a tree which never branches) and usually take up much more space than they need to. The remedy for this is of course to make the formatting smart enough to recognise this situation when it occurs and flexible enough to format the text is a more suitable manner.

An example of this is that if a document contains the index entries[1]

```
\index{Bernoulli!Jacob}\index{Bernoulli!Johann}
```

then it is probably reasonable to format this as

Bernoulli,
  Jacob
  Johann

but if the index entries instead were

```
\index{Jacobi!Carl}\index{Bernoulli!Jacob}
```

then it is probably better to format this as

Bernoulli, Jacob
Jacobi, Carl

than as

Bernoulli,
  Jacob
Jacobi,
  Carl

The makeindex program has some features in this direction, but they only allow dependence on the previous item in the index, not the next item, which is what you need to know when deciding whether 'Jacob' should go on the same line as 'Bernoulli'. Therefore docindex pretty much ignores these features in makeindex and instead sees to that each command that is to typeset an index item knows what kind of items were before and after it.

Another reason for writing this package was to try out the template mechanisms as provided by the LaTeX 2ε* template package.[2] My impression is that this experiment turned out strikingly well. I have always found the more layout-oriented aspects of TeX programming a bit cumbersome, but the separation of layout details from control structures that becomes natural when employing template mechanisms seems to have made it much easier. I'm not sure why this is so, but it could be as simple as that the layout settings are no longer diluted in the control structures. In any case, I would recommend people creating new LaTeX 2ε packages to employ template mechanisms in at least the initial development versions of the package, for the following reasons: (i) it reduces the work of updating the package for LaTeX 2ε*, (ii) it furthers the development of LaTeX 2ε*, and (iii) it might actually become a better package.

A third reason for writing the docindex package was to get the LaTeX document "back in control" of how the index is formatted. Certainly it is the document which has the final say about what the command in the `.ind` file actually do,

---

[1] I'm using the default makeindex metacharacters in these examples, but the style file provided for this package actually uses the same metacharacters as those style files provided by the doc package—hence the 'doc' in 'docindex'.

[2] LaTeX 2ε* is the name of the LaTeX version after LaTeX 2ε. Rather than being a completely different kernel/format, LaTeX 2ε* is (will be) implemented as a collection of LaTeX 2ε packages which replace parts of the kernel. More information and package code can be found on the LaTeX-project website [6].

but the traditional makeindex style files that are used place severe restrictions on the formatting of the index simply because they control where the commands are put. docindex tries to reduce these restrictions by making all texts in the index arguments of commands. Certainly there is a lot more that could be done in this direction—in particular, the (page) numbers in the index could be coded as a `\do`-type list rather than as an explicitly comma-separated list as is done now—but what is in docindex at the moment seems to satisfy all my current needs.

2 Usage

2.1 Straightforward usage

To make use of the docindex package in formatting the index and list of changes of a doc-type LaTeX document, you must do the following:

1. Load the docindex package (or probably rather the docidx2e package—see below).
2. Make sure that the index entries does not use any commands, such as `\verb`, that rely on changing catcodes or otherwise need to be executed before the entire entry text has been tokenized.
3. Generate the `.ind` and `.gls` files using `docindex.ist` as style file for makeindex.

(Item 2 may seem like a monumental task if one considers what the indices of doc documents traditionally looks like—there's a `\verb` for every macro name in the index—but it is really not that bad. docindex loads the xdoc package [4] which redefines `macro`, the cross-referencing mechanism, etc. so that the index entries generated by these no longer uses `\verb`. What is left for you to deal with are merely the possible uses of `\verb` in explicit `\index` or `\SortIndex` commands.)

What advantages are there then for the normal user in having docindex formatting the index and list of changes, as opposed to using the default mechanisms in the doc package? I can only think of two: the index or list of changes may be typeset in a single column and the same makeindex style file can be used for both index and list of changes. Neither advantage is significant. Instead the advantage of docindex lies in that it becomes much simpler to change the formatting, which is rather an advantage for advanced users which have special needs, and in particular one can do this without having to supply e.g. extra makeindex style files.

Another important point is that what you will want to use is probably not the LaTeX $2_{\varepsilon}*$ docindex package, but the "downgraded" LaTeX 2_{ε} version docidx2e, as the former uses the galley2 package which currently wrecks pretty much all justification in all existing document classes. docidx2e provides the same features as docindex, but configuring it is somewhat more cumbersome since template won't do most of the coding for you. It is however rather straightforward to convert a definition using the docindex package to something which achieves the same results with the docidx2e package.

2.2 Multiple indices

The docindex package makes it comparatively simple to include several indices in the same document: all one has to do is use an instance or template of type `docindex` for each index one wishes to typeset. The syntax for using such an instance is

```
\UseInstance{docindex}{⟨instance⟩}{⟨prologue⟩}{⟨epilogue⟩}
```

The ⟨*prologue*⟩ and ⟨*epilogue*⟩ are texts which will be printed just before and after the index, respectively, and either may be empty. The text for the index itself is read from another file, the name and extentsion of which are specified by the instance. The `std` template prints the ⟨*prologue*⟩ and ⟨*epilogue*⟩ in one-column mode, whereas the index itself can be printed in one- or multicolumn mode (the default is three columns).

The doc commands `\PrintIndex` and `\PrintGlossary` are redefined to be

```
\UseInstance{docindex}{index}{\index@prologue}{}
```

and

```
\UseInstance{docindex}{changes}{\glossary@prologue}{}
```

respectively. The `index` and `changes` instances of type `docindex` give the same formatting as the doc defaults. (The docidx2e definitions even use the doc package parameters where applicable, but in docindex it is much simpler to redefine the instance from scratch.)

The format of the sorted index files (`.ind`, `.gls`, etc.) that a `docindex` instance inputs is rather complicated and I would suggest that the generation of these files is left to the makeindex program, but the complete syntax is described in Subsection 3.3. The syntax of the unsorted index files (`.idx`, `.glo`, etc.) is simpler; there are only a few things that are different from the index files of the doc package.

The foremost difference is that the index entries should begin not with `\indexentry` or `\glossaryentry`, but with `\docindexentry`. The xdoc package provides hooks with which one can change these texts in entries generated using the `\index` and `\glossary` commands (as well as higher-level commands built on these, such as the `\SortIndex` and `\changes` commands) and docindex will use these hooks unless it gets passed the `oldkeywords` option. If you are creating a third unsorted index file then you will have to make sure that the command for writing to that file uses `\docindexentry` in the right place.

`oldkeywords` option

The other difference concerns the composite page numbers. The string which separates the parts of a composite page number is not a hyphen '`-`', but the string '`\+`'. (The `\+` command is locally defined for the typesetting of each index by the `docindex` template being used, and the default is to typeset a hyphen.) Again the xdoc package provides a hook for this, and this hook is used by docindex unless it gets passed the `oldkeywords` option.

It also deserves to be listed which the metacharacters are that are the same as in doc indices. The level separator is '`>`', the sort key/item text separator is '`=`', and the quote character is '`!`'. All other makeindex metacharacter parameters have their default values.

2.3 Configuration

Configuration of the layout provided by the docindex package is primarily done by redefining the `index` and `changes` instances of type `docindex`, since these are the instances that are used by the `\PrintIndex` and `\PrintChanges` commands.

The index in the `source2e.tex` file (the main driver for the LaTeX 2_ε source) differs from the default in three respects: it is set in two columns rather than three, there is no seperator character between the parts of a composite page number, and the pagestyle is set to `docindex` during the index. This is set up by the redefinition

```
\DeclareInstance{docindex}{index}{std}{
   columns=2, page-compositor={}, pagestyle=docindex
}
```

(There are however also some changes of parameters related to linebreaking; more on that in connection to configuration of the `changes` instance below.)

Another kind of modification can be found in the tclldoc package [3]. Here the primary level in the index is used for names of procedures and variables, whereas the secondary level for the namespace of the same (the same name may have different definitions in different namespaces). If there is only one namespace for a given name then it is probably overkill to format them as two different index items, but better to join them. This can be achieved through the redefinition

```
\DeclareInstance{docindex}{index}{std}{%
   item1=fixed-r1a, item2=aloneaccept2
}%
```

An item handled by the `fixed-r1a` instance (of type `indexitem`) always tries to join with the following item but rejects to join with the preceeding one. An item handled by the `aloneaccept2` instance accepts to join with the preceeding item if neither that nor the following item is a secondary level item. Thus an item for a name will join with the following item for a namespace if there is only one such item. As the reader no doubt realises, this also solves the problem with the Bernoullis that was described in the introduction.

As for configuring the list of changes formatting, it is instructive to start by considering its default definition:

```
\DeclareInstance{docindex}{changes}{std}{
   file-extension = gls,
   item2 = fixed-r2a-nocomma,
   item3 = fixed-a3r,
   columns = 2,
   letter-format = ,
   letter-skip = 0pt
}
```

In the list of changes a secondary level item (which contains the name of the macro or whatever which was changed) is joined with the following tertiary level item (which details the change that was made). There are two columns and letter groups are not given any special formatting.

The definition of `changes` that would be used for `source2e.tex` differs from the above in only one keyval, namely *body-setup*, but that contains quite a lot of material. To begin with there is the default `\small` which selects the font. Then there is a `\makeatletter` which is needed because some `\changes` entries in the LaTeX sources include commands (e.g. `\TeX`) that (when written to file) expand to other commands whose names include the `@` character. If these are to be tokenized correctly, `@` must be a letter when the `.gls` file is being inputted. Last, but not least, there is a modification of the linebreaking parameters:

```
\UseTemplate{linebreak}{TeX}{
```

The file `source2e.tex` explicitly sets *hbadness* and *hfuzz* to make TEX shut up about over- and underfull hboxes.

```
   hbadness=10000, hfuzz=\maxdimen,
```

In addition to this, there are a couple of parameters that are set by the `multicols` environment to values quite different from the defaults of the `TeX` template and thus must be set too. Here they are shown with their default values. The value of *emergencystretch* could probably be increased.

```
   pretolerance=-1, tolerance=9999, emergencystretch=8pt
}
```

Summing that up, we arrive at the following definition of the `changes` instance for `source2e.tex`.

```
\DeclareInstance{docindex}{changes}{std}{
   file-extension = gls,
   item2 = fixed-r2a-nocomma,
   item3 = fixed-a3r,
   columns = 2,
   letter-format = ,
   letter-skip = 0pt,
   body-setup = \small\makeatletter
      \UseTemplate{linebreak}{TeX}{
         hbadness=10000, hfuzz=\maxdimen,
         pretolerance=-1, tolerance=9999, emergencystretch=8pt
      }
}
```

Another example can be found in the fisource package[3] (v 2.10 or later), which sets up formatting for the fontinst source. There the list of changes should be set in one column, with items from the tertiary level being joined with their parent secondary level items iff the tertiary item is the only one having that particular parent item. This is achieved through the definition

```
\DeclareInstance{docindex}{changes}{std}{%
    file-extension = gls,
    item2 = fixed-r2a-nocomma,
    item3 = aloneaccept3,
    columns = 1,
    letter-format = {},
    letter-skip = 0pt
 }
```

where the difference to the default definition is in the values for the *item3* and *columns* keys.

For details on what they various keys mean, see the declaration of the `std` template of type `docindex` on page 22.

With the docidx2e package, configuration follows the same logic, even though it is much more technical as one has to define the instances without the help of a template. The default instance definitions for the docidx2e package are the

---

[3] It should probably rather be made a document class, but I haven't found it that necessary to change that aspect of it.

```
\@namedef{TP@I{}{docindex}{index}}#1#2{...}
\@namedef{TP@I{}{docindex}{changes}}#1#2{...}
```

that begin on pages 25 and 27 respectively.

3 Implementation

3.1 docstrip modules

This file contains a number of docstrip module directives, and many of these guard code that is not going to be used. In part this mirrors the development of the code (and may get cleared up eventually), but most of this duplication has to do with making the code work in many different set-ups (some of which involve other packages whose interface is rapidly changing).

The modules which control LaTeX code are:

pkg Main guard for code that is to end up in some LaTeX package.

template Guard for code which uses features of the template package. This code will end up in the docindex package, whereas the equivalent code which avoids using templates ends up in the docidx2e package.

default This code protects the default values for template keys. The syntax for this is changing, so the default values are currently being assigned in the template bodies instead.

The modules which control makeindex style files are:

ist Code for the main style file `docindex.ist`.

idx Code for a style file which is like the main one, but the input parameters are set to the same values as in the standard LaTeX `gind.ist`.

glo Code for a style file which is like the main one, but the input parameters are set to the same values as in the standard LaTeX `gglo.ist`.

3.2 Initial stuff

```
⟨*pkg⟩
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage
⟨template⟩    {docindex}
⟨!template⟩    {docidx2e}
   [2001/04/11 v1.00 doc index formatting package]
```

Since the `multicols` environment is used by the `std` template of type `docindex`, the multicol package must have been loaded.

```
\RequirePackage{multicol}
```

This will probably change in docindex once I get around to check how this kind of thing is implemented in the LaTeX $2_{\varepsilon}*$ output routine.

Since the `docindex` pagestyle may be used the xdoc package must have been loaded. This also loads the doc package which contains the definition of `\pfill`.

```
\RequirePackage{xdoc2}[2001/03/26]
```

`oldkeywords` option The `oldkeywords` option tells the docindex package to not change the index entry keywords from the doc defaults. The code for this option appears further down.

```
\DeclareOption{oldkeywords}{}
```

`usedocindexps` option The `usedocindexps` option tells the docindex package to set the pagestyle to `docindex` (defined by xdoc) when typesetting the index. The code for this option appears further down.

```
\DeclareOption{usedocindexps}{}
```

```
\ProcessOptions\relax
⟨/pkg⟩
```

3.3 Index style files

The makeindex style files uses four commands. The most important command is `\indexitem` `\indexitem`, which has the two syntaxes

> `\indexitem{`⟨*level*⟩`}{`⟨*text*⟩`}{`⟨*next level*⟩`}`
> `\indexitem{`⟨*level*⟩`}{`⟨*text*⟩`}{9}{`⟨*numbers*⟩`}{`⟨*next level*⟩`}`

The ⟨*level*⟩ is an integer in the range 1–3, the ⟨*next level*⟩ is an integer in the range 0–3, the ⟨*text*⟩ is the item text, and the ⟨*numbers*⟩ is a list of (page or the like) numbers. The reason for this dual syntax is limitations of makeindex: there is no way of making the text inserted after an item depend on whether there are any page numbers for this item, so one cannot make ⟨*numbers*⟩ a straightforward optional argument.

The level numbers specify at what level the item is. Level 1 corresponds to `\item`, level 2 corresponds to `\subitem`, and level 3 corresponds to `\subsubitem`. The ⟨*next level*⟩ number may also be 0, and that denotes non-`\indexitem` material such as a space between letter groups or the end of the index. The purpose of the ⟨*next level*⟩ argument is to let the formatting of an item depend on what level the next item has, a feature that makeindex alone doesn't provide. Since makeindex only supports putting text in front of things, each new item must begin by inserting the closing brace on the second last argument and the very last argument of the *previous* item before it can do anything for itself. This leads to the following contents of the makeindex `item_`... parameters.

```
⟨*ist | idx | glo⟩
item_0  "}{1}\n\\indexitem{1}{"
item_1  "}{2}\n  \\indexitem{2}{"
item_01 "}{2}\n  \\indexitem{2}{"
item_x1 "}{2}\n  \\indexitem{2}{"
item_2  "}{3}\n    \\indexitem{3}{"
item_12 "}{3}\n    \\indexitem{3}{"
item_x2 "}{3}\n    \\indexitem{3}{"
```

```
delim_0 "}{9}{"
delim_1 "}{9}{"
delim_2 "}{9}{"
delim_n ", "
delim_r "--"
⟨/ist | idx | glo⟩
```

`\indexitem`
`\DI@indexitem`
`\DI@indexitem@`
`\DI@last@level`

The `\indexitem` command (and its subsidiary macros `\DI@indexitem` and `\DI@indexitem@` only handle argument grabbing and some elementary processing of level numbers. The formatting of the item is instead handled by the `\DI@indexitem@`⟨*level*⟩, where ⟨*level*⟩ is the roman numeral `i`, `ii`, or `iii`, family of control sequences. `\indexitem` itself doesn't grab any arguments, instead it inserts the contents of `\DI@last@level` as an additional argument in front of `\DI@indexitem`. The actual argument structures of the other macros are

> `\DI@indexitem{`⟨*last*⟩`}{`⟨*this*⟩`}{`⟨*text*⟩`}{`⟨*next/9*⟩`}`
> `\DI@indexitem@{`⟨*cmd*⟩`}{`⟨*last*⟩`}{9}{`⟨*text*⟩`}\NoValue{`⟨*figures*⟩`}{`⟨*next*⟩`}`

where ⟨*this*⟩ is the level of this item, ⟨*next*⟩ is the level of the next item, ⟨*text*⟩ is the item text, and ⟨*figures*⟩ are the (page) numbers for this item. Several of the arguments of `\DI@indexitem@` are immediately gobbled; they are only used when the original `\indexitem` did not have a ⟨*numbers*⟩ argument.

The `\DI@last@level` macro stores the level of the last item before the current. It is set and used by `\DI@indexitem@`.

```
⟨*pkg⟩
\newcommand\indexitem{%
   \relax
   \expandafter\DI@indexitem \expandafter{\DI@last@level}%
}%
```

```
\def\DI@indexitem#1#2#3#4{%
   \edef\DI@last@level{\number#2\expandafter}%
   \ifnum #4=9
      \expandafter\expandafter \expandafter\DI@indexitem@
   \fi
   \csname DI@indexitem@\romannumeral#2\expandafter\endcsname
      {#1}{#4}{#3}\NoValue
}
```

```
\def\DI@indexitem@#1#2#3#4#5#6#7{#1{#2}{#7}{#4}{#6}}
```

```
\def\DI@last@level{0}
⟨/pkg⟩
```

`\DI@indexitem@`⟨*level*⟩

The `\DI@indexitem@`⟨*level*⟩, where ⟨*level*⟩ is the lower case roman numeral form of the level number, family of control sequences have the syntax

> `\DI@indexitem@`⟨*level*⟩ `{`⟨*previous*⟩`}{`⟨*next*⟩`}{`⟨*text*⟩`}{`⟨*figures*⟩`}`

where ⟨*previous*⟩ and ⟨*next*⟩ are the levels of the previous and following index items, ⟨*text*⟩ is the entry text of this item, and ⟨*figures*⟩ are the (page) numbers for this item, if it has any, or the token `\NoValue`, if it hasn't.

```
⟨*ist | idx | glo⟩
group_skip      "}{0}\n%^^A\n\\indexnewletter{0}{"
heading_prefix  ""
heading_suffix  ""
headings_flag   1
⟨/ist | idx | glo⟩
```

`\indexnewletter`

The `\indexnewletter` command is placed in front of a new letter group. It has the syntax

```
\indexnewletter{⟨first⟩}{⟨letter⟩}{⟨next⟩}
```

where ⟨*first*⟩ is a flag (`1` if this `\indexnewletter` is at the very beginning of the index, `0` otherwise), ⟨*letter*⟩ is the letter name (according to the makeindex program; it can be e.g. the string 'Symbols') and ⟨*next*⟩ is the level of the next item (I think this will always be `1` with makeindex). The command takes care of declining an offer to join with the previous index item, inserts some vertical space if the ⟨*first*⟩ is `0`, print the ⟨*letter*⟩ using `\DI@letter@format`, and doesn't offer to join with the following item.

```
⟨*pkg⟩
\@ifundefined{indexnewletter}{}{%
   \PackageInfo
⟨template⟩       {docindex}
⟨!template⟩       {docidx2e}
      {Command \protect\indexnewletter\space redefined}
}
\outer\def\indexnewletter#1#2#3{%
   \DI@item@nojoin
   \ifnum #1=\z@ \vspace{\DI@letter@skip}\fi
   \DI@letter@format{#2}%
   \def\DI@last@level{0}%
   \let\DI@item@join\@firstofone
   \let\DI@item@nojoin\@empty
}
⟨/pkg⟩
```

The index style files also need to set some parameters which aren't directly connected to the commands provided by the docindex package. First there's the input style parameters:

```
⟨*ist | idx | glo⟩
actual '='
quote '!'
level '>'
```

Then the page precedence should be changed. This is mainly for the convenience of use with documents that `\DocInclude` files, since these by default number the files using letters.

```
page_precedence  "naArR"
```

In `docindex.ist`, both the `keyword` and the `page_compositor` strings are different from their standard values. It turns out to be hard to use a normal command as page compositor, because makeindex always rejects spaces and braces in the page number even if they is in the `page_compositor` parameter!

```
⟨ist⟩keyword "\\xdocindexentry"
⟨ist⟩page_compositor "\\+"
```

Finally, in the style file for the list of changes, the keyword must be changed to `\glossaryentry`.

```
⟨glo⟩keyword "\\glossaryentry"
⟨/ist | idx | glo⟩
```

`oldkeywords` option
`\XD@index@keyword`
`\XD@glossary@keyword`
`\XD@page@compositor`

To make the contents of the `.idx` and `.glo` files compatible with the input parameter settings of `docindex.ist`, some macros used by the xdoc package must be redefined. This can however be stopped if the `oldkeywords` option is passed to the docindex package.

```
⟨*pkg⟩
\@ifpackagewith
⟨template⟩    {docindex}
⟨!template⟩    {docidx2e}
   {oldkeywords}{}{
   \edef\XD@index@keyword{\@backslashchar xdocindexentry}
   \let\XD@glossary@keyword\XD@index@keyword
   \def\XD@page@compositor{\@backslashchar +}
}
```

`\docindexguard`
`\DI@ind@setup`

The first line of every `docindex` style sorted index file is

```
\docindexguard{\endinput}
```

If the index file is inputted as a classical sorted index file then this will produce an undefined command error and no more lines in the index will be read. If the index file is inputted using the conventions of the docindex package then the `\docindexguard` will instead gobble the `\endinput` so that the file will be read.

One can also have the opposite problem: a classical style index file is being input using docindex conventions. It is to overcome this problem that the `\DI@ind@setup` command has been introduced. Classical style index files begin by a `\begin` command, so that command is temporarily redefined to print a warning message and `\endinput` the file. Should the first command instead be `\docindexguard` then everything will be reset to normal. To accomplish this, `\DI@ind@setup` opens a group which should be closed by the initial `\docindexguard` or `\begin`.

```
\def\DI@ind@setup{\bgroup
   \def\docindexguard##1{\egroup}%
   \def\begin##1{%
      \egroup
      \PackageWarningNoLine
⟨template⟩          {docindex}%
⟨!template⟩          {docidx2e}%
         {Ignoring old style index file}
      \endinput
   }%
}
⟨/pkg⟩
```

```
⟨*idx | glo | ist⟩
preamble  "\\docindexguard{\\endinput}\n%^^A\n\\indexnewletter{1}{"
postamble "}{0}\n\\endinput"
⟨/idx | glo | ist⟩
```

In summary, this is the BNF syntax for a sorted index file that is to be typeset using docindex:

⟨*sorted index file*⟩ ⟶ ⟨*guard*⟩⟨*lettergroups*⟩`\endinput`
⟨*guard*⟩ ⟶ `\docindexguard{\endinput}`
⟨*lettergroups*⟩ ⟶ ⟨*lettergroup*⟩ | ⟨*lettergroup*⟩⟨*lettergroups*⟩
⟨*lettergroup*⟩ ⟶ ⟨*heading*⟩⟨*items*⟩

⟨*heading*⟩ ⟶ `\indexnewletter{`⟨*first*⟩`}{`⟨*letter*⟩`}{`⟨*next*⟩`}`
⟨*items*⟩ ⟶ ⟨*empty*⟩ | ⟨*item*⟩⟨*items*⟩
⟨*item*⟩ ⟶ `\indexitem{`⟨*level*⟩`}{`⟨*text*⟩`}{`⟨*next*⟩`}` |
    `\indexitem{`⟨*level*⟩`}{`⟨*text*⟩`}{9}{`⟨*numbers*⟩`}{`⟨*next*⟩`}`

A ⟨*level*⟩ is `1`, `2`, or `3`. A ⟨*next*⟩ is `0`, `1`, `2`, or `3`. Within a ⟨*lettergroup*⟩, the ⟨*next*⟩ of one ⟨*item*⟩ or the ⟨*heading*⟩ must equal the ⟨*level*⟩ of the next ⟨*item*⟩ and the ⟨*next*⟩ of the last item must be `0`. The ⟨*first*⟩ should be `1` in the first ⟨*lettergroup*⟩ and `0` in all the others.

3.4 Template mechanisms

The docindex package loads the xhj and galley2 packages to gain access to the `justification` type templates. This indirectly loads the xparse and template packages.

```
⟨*pkg⟩
⟨template⟩\RequirePackage{xhj,galley2}
```

Since the docidx2e package doesn't use the template mechanisms provided by the template package, but still is to follow the logic of the docindex package which does use these mechanisms, it becomes convenient to define fakes for a couple of template commands. First docidx2e checks if the real template package has been loaded and emits a warning if it has.

```
⟨*!template⟩
\@ifpackageloaded{template}{
   \PackageWarningNoLine{docidx2e}{The docidx2e package is only meant%
      \MessageBreak for use when LaTeX2e* packages like
      template\MessageBreak are not available.}
}{}
```

Before continuing with the definitions, some of the data structures used by the template mechanisms must be explained. A template *instance* is really only a macro; what makes instances different from macros in general is that they usually aren't explicitly programmed. Instead they are formed by combining two different pieces of code: one which is the code part of some template, and one which is a piece of code which sets the *container* macros/registers/parameters for the key values of this template. In general, the first piece of code contains the programming-like aspects of what the instance does, whereas the latter contains those that have to do with lauout and design. The advantage of this model is that it lets you implement many layouts without requiring you to know everything about LaTeX programming that it would take to implement everything using macros.

Instances are stored in control sequences of the form

`\TP@I{`⟨*collection*⟩`}{`⟨*type*⟩`}{`⟨*name*⟩`}`

`\TP@I{`⟨*collection*⟩`}{`⟨*type*⟩`}{`⟨*name*⟩`}`

The ⟨*type*⟩ is the primary distinction between instances; for each type there exists a specification of what all instances of that type must do, and all instances of a type must be interchangeable. In particular, all instances of a given template type must have the same argument structure. The ⟨*name*⟩ is simply the name used to identify the instance (amongst all other instances of that type). Finally, the ⟨*collection*⟩ is something which can be used in circumstances where one needs to quickly switch between different definitions of an instance. If they have different ⟨*collection*⟩s then they can exist simutaneously; which of them is used is determined by which collection is currently active.

`\TP@T{`⟨*type*⟩`}`

Collections are active on a "per type" basis; which collection is active for instances of type ⟨*type*⟩ is determined by the contents of the `\TP@T{`⟨*type*⟩`}` control sequences, which are macros with the structure

`{`⟨*collection*⟩`}{`⟨*arguments*⟩`}`

If there is no instance with the requested name in the currently active collection then the instance with the same name from the normal collection (whose name is the empty string) will be used. The ⟨*arguments*⟩ part of the macro is simply the number of arguments of instances of this type; it is only used when declaring templates.

`\UseCollection`

The `\UseCollection` command sets the active collection for a given type. It has the syntax

`\UseCollection{`⟨*type*⟩`}{`⟨*collection*⟩`}`

This macro was used up to v 1.00 of docindex but a change in the package logic made it unnecessary.

```
% \providecommand*\UseCollection[2]{%
%    \expandafter\edef \csname TP@T{#1}\endcsname{%
%       {#2}%
%       {\expandafter\expandafter \expandafter\@secondoftwo
%          \csname TP@T{#1}\endcsname}%
%    }%
% }
```

`\@letinstance`

The `\@letinstance` macro `\let`s the (currently used) instance with given name and type to the ⟨*target*⟩ control sequence. It has the syntax

`\@letinstance{`⟨*target*⟩`}{`⟨*type*⟩`}{`⟨*name*⟩`}`

```
\def\@letinstance#1#2#3{%
   \expandafter\let \expandafter#1%
      \csname TP@I%
         {\expandafter\expandafter \expandafter\@firstoftwo
            \csname TP@T{#2}\endcsname}%
         {#2}{#3}%
      \endcsname
   \ifx \relax#1%
      \expandafter\let \expandafter#1\csname TP@I{}{#2}{#3}\endcsname
   \fi
}
```

`\UseInstance`

The `\UseInstance` calls the (currently used) instance with given name and type. Its syntax is

`\UseInstance{`⟨*type*⟩`}{`⟨*name*⟩`}` ⟨*arguments of instance*⟩

```
\providecommand*\UseInstance[2]{%
   \@letinstance\@tempa{#1}{#2}%
   \ifx \relax\@tempa
      \PackageError{docidx2e}{Instance #2 of type #1 undefined}\@eha
```

```
    \else
       \expandafter\@tempa
    \fi
}
⟨/!template⟩
```

3.5 Templates for index item formatting

`justification` type

In docidx2e, we have to provide a dummy definition of `\TP@T{justification}`.

```
⟨!template⟩\@namedef{TP@T{justification}}{{}{0}}
```

`justification/indexitem1` instance
`justification/indexitem2` instance
`justification/indexitem3` instance

The `indexitem`⟨*level*⟩ instances of the `justification` template set up paragraph indentation etc. for a paragraph containing an index item at that level. The layout is the same as that used by the doc package, but it is not specified in quite the same way.

```
⟨*template⟩
\DeclareInstance{justification}{indexitem1}{single}{
   leftskip=30pt, rightskip=15pt, startskip=-30pt, parfillskip=-15pt,
   linefillskip=0pt plus 1fil, parindent=0pt
}
\DeclareInstance{justification}{indexitem2}{single}{
   leftskip=30pt, rightskip=15pt, startskip=-15pt, parfillskip=-15pt,
   linefillskip=0pt plus 1fil, parindent=0pt
}
\DeclareInstance{justification}{indexitem3}{single}{
   leftskip=30pt, rightskip=15pt, startskip=-5pt, parfillskip=-15pt,
   linefillskip=0pt plus 1fil, parindent=0pt
}
⟨/template⟩
⟨*!template⟩
\@namedef{TP@I{}{justification}{indexitem1}}{%
   \leftskip=30\p@
   \rightskip=15\p@
   \parindent=-30\p@
   \parfillskip=-\rightskip
}
\@namedef{TP@I{}{justification}{indexitem2}}{%
   \leftskip=30\p@
   \rightskip=15\p@
   \parindent=-15\p@
   \parfillskip=-\rightskip
}
\@namedef{TP@I{}{justification}{indexitem3}}{%
   \leftskip=30\p@
   \rightskip=15\p@
   \parindent=-5\p@
   \parfillskip=-\rightskip
}
⟨/!template⟩
```

3.5.1 The `indexitem` template type

`indexitem` type
`\DI@item@nojoin`
`\DI@item@join`

The argument structure of a template of type `indexitem` is

`{`⟨*previous*⟩`}{`⟨*next*⟩`}{`⟨*text*⟩`}{`⟨*figures*⟩`}`

⟨*previous*⟩ and ⟨*next*⟩ are the level codes of the index item before and after the current item, ⟨*text*⟩ is the item text of the current index item, and ⟨*figures*⟩ are the (page) numbers for this item, if it has any, or the token `\NoValue`, if it hasn't.

Templates of this type format and typeset one item in an index. In doing so they may do pretty much anything as long as the other items aren't affected: they may start and end paragraphs, change the paragraph justification, …

There is however one area in which the rules are rather strict, and that has to do with when two items can be joined. In a case where item A is followed by item B, item A can propose to item B that they should be joined and item B can then accept or decline this offer. Technically the offer consists of defining the two macros `\DI@item@join` and `\DI@item@nojoin`. If item B accepts the offer it will execute `\DI@item@join` and if it declines the offer it will execute `\DI@item@nojoin`. A typical definition of `\DI@item@join` might be to insert a punctuation mark and a typical definition of `\DI@item@nojoin` might be to end the current paragraph.

There is however also a third case, namely that no offer was given. In this case `\DI@item@nojoin` should be `\let` to `\@empty` and `\DI@item@join` should be `\let` to `\@firstofone`. The reason for this last rule is that `\DI@item@join` has the syntax

`\DI@item@join{`⟨*no-join recovery code*⟩`}`

where the ⟨*no-join recovery code*⟩ is code that item B needs to have executed if there is no join although item B would have accepted it. `\DI@item@nojoin`, on the other hand, takes no argument.

```
⟨template⟩\DeclareTemplateType{indexitem}{4}
⟨!template⟩\@namedef{TP@T{indexitem}}{{}{4}}
\let\DI@item@join=\@firstofone
\let\DI@item@nojoin=\@empty
```

`indexitem/fixed` template

The `fixed` template of type `indexitem` formats an item as the items in `doc`'s `theindex` environment. It is fixed in that it ignores the levels of the surrounding items.

The keys for this template are:

justification-setup (i) This is a template instance of type `justification`. It sets the justification for the paragraph containing the item, unless the item is being joined with the preceeding item.

pre-join (b) A switch for whether the item should accept to be joined with the item before. True means "accept", false means "decline" (which is the default).

nofig-action (f1) If the ⟨*figures*⟩ argument is `\NoValue` then the ⟨*text*⟩ argument is passed on to this macro for the actual formatting. The default expansion is precisely the ⟨*text*⟩.

fig-action (f2) If the ⟨*figures*⟩ argument is not `\NoValue` then the ⟨*text*⟩ and ⟨*figures*⟩ arguments are passed on (in that order) to this macro for the actual formatting. The default expansion is

⟨*text*⟩`\pfill`⟨*figures*⟩

post-join (b) A switch for whether the item should offer to join with the following item. True means “make offer”, false (which is the default) means “don’t make offer”. Making the offer is furthermore conditioned by that the ⟨*figures*⟩ argument is `\NoValue`.

nojoin-extra (f0) Extra code which is inserted after the normal code for an item if the item neither has any figures nor offers to join with the following item. The default value is a space of length *linefillskip* followed by a `\nopagebreak`.

join-extra (f0) Extra text which is inserted after the normal text of the item if there is a join, by default a comma and a space.

offjoin-extra (f0) Extra code which is inserted after the normal text of the item if a join is offered but declined. The default value is a space of length *linefillskip* followed by a `\nopagebreak` (larger than the one from *nojoin-extra*; if not for this, the default could have been taken to be `\DI@nojoin@extra`).

Note that the contents of the *nojoin-extra*, *join-extra*, and *offjoin-extra* keys must be robust as they may be subjected to a `\protected@edef`.

```
⟨*template⟩
\DeclareTemplate{indexitem}{fixed}{4}{
   justification-setup =i{justification}      \DI@item@justification,
   pre-join      =b
⟨default⟩         {false}
                                               DI@prejoin@,
   nofig-action  =f1
⟨default⟩         {#1}
                                              \DI@nofig,
   fig-action    =f2
⟨default⟩         {#1\pfill#2}
                                              \DI@hasfig,
   post-join     =b
⟨default⟩         {false}
                                               DI@postjoin@,
   nojoin-extra    =f0
⟨default⟩         {\hspace*{\justification@g}
⟨default⟩           \protect\nopagebreak[2]}
                                              \DI@nojoin@extra,
   join-extra    =f0
⟨default⟩         {,\space}
                                              \DI@join@extra,
   offjoin-extra =f0
⟨default⟩         {\hspace*{\justification@g}
⟨default⟩           \protect\nopagebreak[4]}
                                              \DI@offjoin@extra
}{%
⟨*!default⟩
   \let\ifDI@prejoin@\iffalse
   \let\DI@nofig\@firstofone
   \def\DI@hasfig##1##2{##1\pfill##2}%
   \let\ifDI@postjoin@\iffalse
   \def\DI@nojoin@extra{%
       \hspace*{\justification@g}\protect\nopagebreak[2]%
   }%
   \def\DI@join@extra{,\space}%
   \def\DI@offjoin@extra{%
      \hspace*{\justification@g}\protect\nopagebreak[4]%
   }%
⟨/!default⟩
   \DoParameterAssignments
   \ifDI@prejoin@
      \DI@item@join{\DI@item@justification}%
   \else
      \DI@item@nojoin\DI@item@justification
   \fi
   \let\DI@item@join\@firstofone
   \let\DI@item@nojoin\@empty
   \IfNoValueTF{#4}{%
      \DI@nofig{#3}%
      \ifDI@postjoin@
         \protected@edef\DI@item@join##1{\DI@join@extra}%
         \protected@edef\DI@item@nojoin{\DI@offjoin@extra\protect\par}%
      \else
         \DI@nojoin@extra\par
      \fi
   }{%
      \DI@hasfig{#3}{#4}%
      \par
   }%
   \ignorespaces
}
```

`indexitem/fixed1` instance
`indexitem/fixed2` instance
`indexitem/fixed3` instance

The `fixed1`, `fixed2`, and `fixed3` instances of type `indexitem` are simply the `fixed` template with different values assigned to the *justification-setup* key.

```
\DeclareInstance{indexitem}{fixed1}{fixed}
   {justification-setup = indexitem1}
\DeclareInstance{indexitem}{fixed2}{fixed}
   {justification-setup = indexitem2}
\DeclareInstance{indexitem}{fixed3}{fixed}
   {justification-setup = indexitem3}
⟨/template⟩
⟨*!template⟩
\@namedef{TP@I{}{indexitem}{fixed1}}#1#2#3#4{%
   \@letinstance\DI@item@justification{justification}{indexitem1}%
   \DI@item@nojoin
   \DI@item@justification
   \ifx \NoValue#4%
      #3\nobreak\hfil\nopagebreak[2]%
   \else
      #3\pfill#4%
   \fi
   \let\DI@item@join\@firstofone
   \let\DI@item@nojoin\@empty
   \par
}
\@namedef{TP@I{}{indexitem}{fixed2}}#1#2#3#4{%
   \@letinstance\DI@item@justification{justification}{indexitem2}%
   \DI@item@nojoin
```

```
  \DI@item@justification
  \ifx \NoValue#4%
    #3\nobreak\hfil\nopagebreak[2]%
  \else
    #3\pfill#4%
  \fi
  \let\DI@item@join\@firstofone
  \let\DI@item@nojoin\@empty
  \par
}
\@namedef{TP@I{}{indexitem}{fixed3}}#1#2#3#4{%
  \@letinstance\DI@item@justification{justification}{indexitem3}%
  \DI@item@nojoin
  \DI@item@justification
  \ifx \NoValue#4%
    #3\nobreak\hfil\nopagebreak[2]%
  \else
    #3\pfill#4%
  \fi
  \let\DI@item@join\@firstofone
  \let\DI@item@nojoin\@empty
  \par
}
⟨/!template⟩
```

`indexitem/fixed-r1a` instance
`indexitem/fixed-r2a-nocomma` instance
`indexitem/fixed-a3r` instance

The `fixed-r1a`, `fixed-r2a-nocomma`, and `fixed-a3r` instances of type `indexitem` are again based on the `fixed` template, but here they always accept (or offer) to join with one neighbouring item, whereas they always reject to join with the other. As before, they differ in their values of the *justification-setup* key, and the `-nocomma` is because that instance only inserts a space, not a comma and a space, when items are joined.

```
⟨*template⟩
\DeclareInstance{indexitem}{fixed-r1a}{fixed}
  {justification-setup = indexitem1, post-join = true}
\DeclareInstance{indexitem}{fixed-r2a-nocomma}{fixed}
  {justification-setup = indexitem2,
   post-join = true, join-extra = {\space}}
\DeclareInstance{indexitem}{fixed-a3r}{fixed}
  {justification-setup = indexitem3, pre-join = true}
⟨/template⟩
⟨*!template⟩
\@namedef{TP@I{}{indexitem}{fixed-r1a}}#1#2#3#4{%
  \@letinstance\DI@item@justification{justification}{indexitem1}%
  \DI@item@nojoin
  \DI@item@justification
  \ifx \NoValue#4%
    #3%
    \def\DI@item@join##1{, }%
    \def\DI@item@nojoin{\nobreak\hfil\nopagebreak[4]\par}%
  \else
    #3\pfill#4\par
    \let\DI@item@join\@firstofone
    \let\DI@item@nojoin\@empty
  \fi
```

```
   \ignorespaces
}
\@namedef{TP@I{}{indexitem}{fixed-r2a-nocomma}}#1#2#3#4{%
   \@letinstance\DI@item@justification{justification}{indexitem2}%
   \DI@item@nojoin
   \DI@item@justification
   \ifx \NoValue#4%
      #3%
      \def\DI@item@join##1{ }%
      \def\DI@item@nojoin{\nobreak\hfil\nopagebreak[4]\par}%
   \else
      #3\pfill#4\par
      \let\DI@item@join\@firstofone
      \let\DI@item@nojoin\@empty
   \fi
   \ignorespaces
}
\@namedef{TP@I{}{indexitem}{fixed-a3r}}#1#2#3#4{%
   \@letinstance\DI@item@justification{justification}{indexitem3}%
   \DI@item@join{\DI@item@justification}%
   \ifx \NoValue#4%
      #3\hfil\nopagebreak[2]%
   \else
      #3\pfill#4%
   \fi
   \let\DI@item@join\@firstofone
   \let\DI@item@nojoin\@empty
   \par
}
⟨/!template⟩
```

`indexitem/aloneaccept`
template

The `aloneaccept` template of type `indexitem` formats an item as the items in doc's `theindex` environment. It accepts to be joined with the preceeding item if and only if both that and the following item are at a lower level than the item itself is.

The keys for this template are:

justification-setup (i) This is a template instance of type `justification`. It sets the justification for the paragraph containing the item, unless the item is being joined with the preceeding item.

ownlevel (C) This is the (nominal) level of this item; it will accept a join with the preceeding item if and only if the levels of both that and the following item are different from this number. The default is 2.

nofig-action (f1) If the ⟨*figures*⟩ argument is `\NoValue` then the ⟨*text*⟩ argument is passed on to this macro for the actual formatting. The default expansion is the ⟨*text*⟩ followed by a space of linefillskip.

fig-action (f2) If the ⟨*figures*⟩ argument is not `\NoValue` then the ⟨*text*⟩ and ⟨*figures*⟩ arguments are passed on (in that order) to this macro for the actual formatting. The default expansion is

⟨*text*⟩`\pfill`⟨*figures*⟩

post-join (b) A switch for whether the item should offer to join with the following item. True means "make offer", false (which is the default) means "don't make offer". Making the offer is furthermore conditioned by that the ⟨*figures*⟩ argument is `\NoValue`.

nojoin-extra (f0) Extra code which is inserted after the normal code for an item if the item neither has any figures nor offers to join with the following item. The default value is a space of length *linefillskip*.

join-extra (f0) Extra text which is inserted after the normal text of the item if there is a join, by default a comma and a space.

offjoin-extra (f0) Extra code which is inserted after the normal text of the item if a join is offered but declined, by default the *nojoin-extra* code followed by a `\nopagebreak`.

Note that the contents of the *nojoin-extra*, *join-extra*, and *offjoin-extra* keys must be robust as they may be subjected to a `\protected@edef`.

```
⟨*template⟩
\DeclareTemplate{indexitem}{aloneaccept}{4}{
  justification-setup =i{justification}       \DI@item@justification,
  ownlevel       =C
⟨default⟩         {2}
                                              \DI@this@level,
  nofig-action   =f1
⟨default⟩         {#1}
                                              \DI@nofig,
  fig-action     =f2
⟨default⟩         {#1\pfill#2}
                                              \DI@hasfig,
  post-join      =b
⟨default⟩         {false}
                                               DI@postjoin@,
  nojoin-extra     =f0
⟨default⟩         {\hspace*{\justification@g}}
                                              \DI@nojoin@extra,
  join-extra     =f0
⟨default⟩         {,\space}
                                              \DI@join@extra,
  offjoin-extra =f0
⟨default⟩         {\DI@nojoin@extra\protect\nopagebreak[4]}
                                              \DI@offjoin@extra
}{%
⟨*!default⟩
  \def\DI@this@level{2}%
  \let\DI@nofig\@firstofone
  \def\DI@hasfig##1##2{##1\pfill##2}%
  \let\ifDI@postjoin@\iffalse
  \def\DI@nojoin@extra{\hspace*{\justification@g}}%
  \def\DI@join@extra{,\space}%
  \def\DI@offjoin@extra{\DI@nojoin@extra\protect\nopagebreak[4]}%
⟨/!default⟩
  \DoParameterAssignments
  \ifnum \DI@this@level=#1
```

```
      \DI@item@nojoin \DI@item@justification
   \else\ifnum \DI@this@level=#2
      \DI@item@nojoin \DI@item@justification
   \else
      \DI@item@join{\DI@item@justification}%
   \fi\fi
   \let\DI@item@join\@firstofone
   \let\DI@item@nojoin\@empty
   \IfNoValueTF{#4}{%
      \DI@nofig{#3}%
      \ifDI@postjoin@
         \protected@edef\DI@item@join##1{\DI@join@extra}%
         \protected@edef\DI@item@nojoin{\DI@offjoin@extra\protect\par}%
      \else
         \DI@nojoin@extra \par
      \fi
   }{%
      \DI@hasfig{#3}{#4}%
      \par
   }%
   \ignorespaces
}
⟨/template⟩
```

`indexitem/aloneaccept2` instance

`indexitem/aloneaceept3` instance

The `aloneaccept2` and `aloneaccept3` instances of type `indexitem` are simply the `aloneaccept` template with the levels fixed to two and three, respectively.

```
⟨*template⟩
\DeclareInstance{indexitem}{aloneaccept2}{aloneaccept}
   {justification-setup = indexitem2, ownlevel = 2}
\DeclareInstance{indexitem}{aloneaccept3}{aloneaccept}
   {justification-setup = indexitem3, ownlevel = 3}
⟨/template⟩
⟨*!template⟩
\@namedef{TP@I{}{indexitem}{aloneaccept2}}#1#2#3#4{%
   \@letinstance\DI@item@justification{justification}{indexitem2}%
   \ifnum #1=\tw@
      \DI@item@nojoin \DI@item@justification
   \else\ifnum #2=\tw@
      \DI@item@nojoin \DI@item@justification
   \else
      \DI@item@join{\DI@item@justification}%
   \fi\fi
   \ifx \NoValue#4%
      #3\nobreak\hfil\vadjust{}%
   \else
      #3\pfill #4%
   \fi
   \let\DI@item@join\@firstofone
   \let\DI@item@nojoin\@empty
   \par
}
\@namedef{TP@I{}{indexitem}{aloneaccept3}}#1#2#3#4{%
   \@letinstance\DI@item@justification{justification}{indexitem3}%
   \ifnum #1=\thr@@
```

```
      \DI@item@nojoin \DI@item@justification
   \else\ifnum #2=\thr@@
      \DI@item@nojoin \DI@item@justification
   \else
      \DI@item@join{\DI@item@justification}%
   \fi\fi
   \ifx \NoValue#4%
      #3\nobreak\hfil\vadjust{}%
   \else
      #3\pfill #4%
   \fi
   \let\DI@item@join\@firstofone
   \let\DI@item@nojoin\@empty
   \par
}
⟨/!template⟩
```

3.5.2 The `docindex` template type

`docindex` type

A template of type `docindex` takes care of typesetting an index found in a file (which is `\input`ted as part of this process), hence using an instance of type `docindex` is the same kind of action that the `\printindex` and `\printglossary` commands make.

The template decides from which file the index should be read. It takes two arguments: the index prologue and the index epilogue. These are two pieces of text (which may just as well include a sectioning command) that are printed just before and after the index. Either argument may be empty. Immediately after the file containing the body of the index has been inputted, the template must execute `\DI@item@nojoin` to make sure that the last item is properly typeset.

Templates of type `docindex` must begin by opening a group and end by closing it. They must furthermore locally define the following macros before any part of the index is typeset.

`\DI@indexitem@i`, `\DI@indexitem@ii`, and `\DI@indexitem@iii` Handlers for index items at level 1, 2, and 3 respectively. These handlers must conform to the specification for `indexitem` instances.

`\DI@letter@skip`, `\DI@letter@format` These are described in the comments to the `\indexnewletter` command.

`\+` The command for typesetting the separator between two parts of a composite (page) number. This is a parameterless macro.

```
⟨template⟩\DeclareTemplateType{docindex}{2}
⟨!template⟩\@namedef{TP@T{docindex}}{{}{2}}
```

`docindex/std` template

The `std` template of type `docindex` typesets an index while providing all the formatting parameters of the `doc` index and list of changes (and a few more).

The keys of the template are:

file-name (n) The base name of the file in which the index is stored, by default the `\jobname`.

file-extension (n) The extension of the file in which the index is stored, by default `ind`.

item1 (i) `indexitem` instance for level 1 items, by default `fixed1`.

item2 (i) `indexitem` instance for level 2 items, by default `fixed2`.

item3 (i) `indexitem` instance for level 3 items, by default `fixed3`.

columns (C) The number of columns in the index, by default 3.

reserved-height (L) The minimal amount of vertical space that must be left on the current page if the index is to start on it, by default 80 pt.

column-sep (l) The horizontal separation between columns in the index, by default 10 pt. (This may seem strange in comparison with doc, since `\IndexParms` contains the command `\columnsep=15pt`, but doc doesn't execute `\IndexParms` until LaTeX is already in multi-column mode, and then it is too late for the changed value to have any effect.)

prologue-setup (f0) Various commands setting layout parameters (e.g. the font) for the prologue; by default empty.

body-setup (f0) Various commands setting layout parameters (e.g. the font) for the body of the index; by default `\small`.

epilogue-setup (f0) Various commands setting layout parameters (e.g. the font) for the epilogue; by default `\normalsize` (to counter the `\small` in the *body-setup*).

letter-skip (L) The skip inserted before a new letter group, by default 10 pt plus 2 pt minus 3 pt.

letter-format (f1) The macro which formats new letter groups; the argument is the heading for the group, as generated by makeindex. By defaults it typesets the argument in boldface, centered on a line.

pagestyle (n) If this is nonempty then the pagestyle by that name will be selected for the index. By default it is empty.

parskip (l) The value of `\parskip` to use inside the index, by default 0 pt plus 1 pt. This key value is likely to change as the LaTeX 2ε∗ interfaces for galleys evolve.

page-compositor (f0) The text that is typeset to separate two parts of a composite (page) number, by default a hyphen.

```
⟨*template⟩
\DeclareTemplate{docindex}{std}{2}{
   file-name         =n
⟨default⟩          {\jobname}
                                                   \DI@file@name,
   file-extension    =n
⟨default⟩          {ind}
                                                   \DI@file@ext,
   item1             =i{indexitem}
⟨default⟩          {fixed1}
                                                   \DI@indexitem@i,
   item2             =i{indexitem}
```

```
⟨default⟩       {fixed2}
                                                    \DI@indexitem@ii,
   item3           =i{indexitem}
⟨default⟩       {fixed3}
                                                    \DI@indexitem@iii,
   reserved-height  =L
⟨default⟩       {80pt}
                                                    \DI@reserved@height,
   columns          =C
⟨default⟩       {3}
                                                    \DI@columns,
   column-sep       =l
⟨default⟩       {10pt}
                                                    \columnsep,
   prologue-setup    =f0
⟨default⟩       {}
                                                    \DI@prologue@setup,
   body-setup        =f0
⟨default⟩       {\small}
                                                    \DI@body@setup,
   epilogue-setup    =f0
⟨default⟩       {\normalsize}
                                                    \DI@epilogue@setup,
   letter-skip      =L
⟨default⟩       {10pt plus 2pt minus 3pt}
                                                    \DI@letter@skip,
   letter-format     =f1
⟨default⟩       {\UseInstance{justification}{center}%
⟨default⟩        \textbf{#1}\nopagebreak\csname par\endcsname}
                                                    \DI@letter@format,
   pagestyle        =n
⟨default⟩           {}
                                                    \DI@pagestyle,
   parskip          =l
⟨default⟩           {0pt plus 1pt}
                                                    \parskip,
   page-compositor  =f0
⟨default⟩           {-}
                                                    \+
}{%
   \begingroup
⟨*!default⟩
   \def\DI@file@name{\jobname}%
   \def\DI@file@ext{ind}%
   \@letinstance\DI@indexitem@i{indexitem}{fixed1}%
   \@letinstance\DI@indexitem@ii{indexitem}{fixed2}%
   \@letinstance\DI@indexitem@iii{indexitem}{fixed3}%
   \def\DI@reserved@height{80pt}%
   \def\DI@columns{3}%
   \columnsep=10pt%
   \let\DI@prologue@setup\@empty
   \def\DI@body@setup{\small}%
   \def\DI@epilogue@setup{\normalsize}%
   \def\DI@letter@skip{10pt plus 2pt minus 3pt}%
   \def\DI@letter@format##1{%
      \UseInstance{justification}{center}%
      \textbf{##1}\nopagebreak\par
   }%
   \parskip=\z@\@plus\p@
   \let\DI@pagestyle\@empty
   \def\+{-}%
⟨/!default⟩
   \DoParameterAssignments
   \IfFileExists{\DI@file@name.\DI@file@ext}{%
      \ifnum \DI@columns>\@ne
         \begin{multicols}{\DI@columns}%
            [\DI@prologue@setup #1][\DI@reserved@height]%
      \else
         \enough@room{\DI@reserved@height}%
         \DI@prologue@setup #1\par
         \addvspace\multicolsep
      \fi
      \ifx \DI@pagestyle\@empty \else \pagestyle{\DI@pagestyle}\fi
      \DI@body@setup
      \DI@ind@setup
      \input{\DI@file@name.\DI@file@ext}%
      \DI@item@nojoin
      \ifx \DI@pagestyle\@empty \else
         \expandafter\thispagestyle \expandafter{\DI@pagestyle}%
      \fi
      \ifnum \DI@columns>\@ne
         \end{multicols}%
      \else
         \enough@room\postmulticols
         \addvspace\multicolsep
      \fi
      \DI@epilogue@setup #2\par
   }{\typeout{No file \DI@file@name.\DI@file@ext.}}%
   \endgroup
}
```

`docindex/index` instance
`usedocindexps` option

The `index` instance of the `docindex` template type prints the normal index (as opposed to the list of changes). There are two different definitions of the instance: one which sets the pagestyle in the index, and one which does not; which one is used depends on whether the `usedocindexps` option has been passed to the package or not.

```
\@ifpackagewith{docindex}{usedocindexps}{%
   \DeclareInstance{docindex}{index}{std}{pagestyle=docindex}%
}{%
   \DeclareInstance{docindex}{index}{std}{}%
}
⟨/template⟩
```

The implementations of the `index` instance in docidx2e are slightly off in that they use doc parameters for various settings in the extent such parameters exist.

```
⟨*!template⟩
\@ifpackagewith{docidx2e}{usedocindexps}{%
   \@namedef{TP@I{}{docindex}{index}}#1#2{%
```

```
      \begingroup
      \@letinstance\DI@indexitem@i{indexitem}{fixed1}%
      \@letinstance\DI@indexitem@ii{indexitem}{fixed2}%
      \@letinstance\DI@indexitem@iii{indexitem}{fixed3}%
      \columnsep=10pt%
      \parskip=0pt plus 1pt%
      \def\DI@letter@skip{10pt plus 2pt minus 3pt}%
      \def\DI@letter@format##1{%
         \par
         \hb@xt@\hsize{\hfil\textbf{##1}\hfil}%
         \nopagebreak
      }%
      \def\+{-}%
      \IfFileExists{\jobname.ind}{%
         \ifnum \c@IndexColumns>\@ne
            \begin{multicols}{\c@IndexColumns}[#1][\IndexMin]%
         \else
            \enough@room{\IndexMin}%
            #1\par
            \addvspace\multicolsep
         \fi
         \pagestyle{docindex}%
         \small
         \@nobreakfalse
         \DI@ind@setup
         \input{\jobname.ind}%
         \DI@item@nojoin
         \thispagestyle{docindex}
         \ifnum \c@IndexColumns>\@ne
            \end{multicols}%
         \else
            \enough@room\postmulticols
            \addvspace\multicolsep
         \fi
         \normalsize #2\par
      }{\typeout{No file \jobname.ind.}}%
      \endgroup
   }
}{%
   \@namedef{TP@I{}{docindex}{index}}#1#2{%
      \begingroup
      \@letinstance\DI@indexitem@i{indexitem}{fixed1}%
      \@letinstance\DI@indexitem@ii{indexitem}{fixed2}%
      \@letinstance\DI@indexitem@iii{indexitem}{fixed3}%
      \columnsep=10pt%
      \parskip=0pt plus 1pt%
      \def\DI@letter@skip{10pt plus 2pt minus 3pt}%
      \def\DI@letter@format##1{%
         \par
         \hb@xt@\hsize{\hfil\textbf{##1}\hfil}%
         \nopagebreak
      }%
      \def\+{-}%
      \IfFileExists{\jobname.ind}{%
```

```
         \ifnum \c@IndexColumns>\@ne
            \begin{multicols}{\c@IndexColumns}[#1][\IndexMin]%
         \else
            \enough@room{\IndexMin}%
            #1\par
            \addvspace\multicolsep
         \fi
         \small
         \@nobreakfalse
         \DI@ind@setup
         \input{\jobname.ind}%
         \DI@item@nojoin
         \ifnum \c@IndexColumns>\@ne
            \end{multicols}%
         \else
            \enough@room\postmulticols
            \addvspace\multicolsep
         \fi
         \normalsize #2\par
      }{\typeout{No file \jobname.ind.}}%
      \endgroup
   }
}
⟨/!template⟩
```

`docindex/changes` instance

The `changes` instance of the `docindex` template type typesets a doc list of changes.

```
⟨*template⟩
\DeclareInstance{docindex}{changes}{std}{
   file-extension = gls,
   item2 = fixed-r2a-nocomma,
   item3 = fixed-a3r,
   columns = 2,
   letter-format = {},
   letter-skip = \z@skip
}
⟨/template⟩
```

```
⟨*!template⟩
\@namedef{TP@I{}{docindex}{changes}}#1#2{%
   \begingroup
   \@letinstance\DI@indexitem@i{indexitem}{fixed1}%
   \@letinstance\DI@indexitem@ii{indexitem}{fixed-r2a-nocomma}%
   \@letinstance\DI@indexitem@iii{indexitem}{fixed-a3r}%
   \columnsep=10pt%
   \parskip=0pt plus 1pt%
   \def\DI@letter@skip{\z@skip}%
   \let\DI@letter@format\@gobble
   \def\+{-}%
   \IfFileExists{\jobname.gls}{%
      \ifnum \c@GlossaryColumns>\@ne
         \begin{multicols}{\c@GlossaryColumns}[#1][\GlossaryMin]%
      \else
         \enough@room{\GlossaryMin}%
         #1\par
         \addvspace\multicolsep
      \fi
      \small
      \makeatletter
      \@nobreakfalse
      \DI@ind@setup
      \input{\jobname.gls}%
      \DI@item@nojoin
      \ifnum \c@GlossaryColumns>\@ne
         \end{multicols}%
      \else
         \enough@room\postmulticols
         \addvspace\multicolsep
      \fi
      \normalsize #2\par
   }{\typeout{No file \jobname.gls.}}
   \endgroup
}
⟨/!template⟩
```

`\PrintIndex`
`\PrintChanges`

The `\PrintIndex` and `\PrintChanges` commands are redefined to use the respective instances of template type docindex.

```
\renewcommand\PrintIndex{%
   \UseInstance{docindex}{index}{\index@prologue}{}%
   \global\let\PrintIndex\@empty
}
\renewcommand\PrintChanges{%
   \UseInstance{docindex}{changes}{\glossary@prologue}{}%
   \global\let\PrintChanges\@empty
}
⟨/pkg⟩
```

4 Notes and acknowledgements

The exact descriptions of the parameters of the makeindex program is the paper [2] by Chen and Harrison, but I have seen claims that there are parameters not listed there (presumably becuase they were added after this paper was written). `docindex.ist` does not change any such undocumented parameter, however.

There are other index sorting programs than makeindex around, such as for example xindy [5]. Should someone create index style files for such systems that are equivalent (or superior, for that matter) to `docindex.ist` then I would be happy to add them to the docindex distribution.

Most of the actual layout parameter settings used by the docindex package are not of my design, but copied from various other LaTeX packages such as [7, 8] (mainly by Frank Mittelbach). I have however tried to sort out which parameters are actually in force under the various conditions—something which turned out to be less obvious than I originally suspected.

The idea to have the `docindex` type templates input the sorted index file (rather than simply setting up the formatting of it as was the case in v 0.03) was taken from the xindex package [1] by Achim Blumensath.

References

[1] Achim Blumensath: *The xindex package*; HTTP://`www-mgi.informatik.rwth-aachen.de/~blume/`.

[2] Pehong Chen, Michael A. Harrison: *Index Preparation and Processing*, Software: practice & experience, vol. **19**, no. 9 (1988), 897–915; Computer Science Tech. Report 87/347, University of California, Berkeley, March 1987; CTAN: `indexing/makeindex/paper/ind.tex`.

[3] Lars Hellström: *The tclldoc package and class*, v 2.20 or newer; CTAN: `macros/latex/contrib/tclldoc/tclldoc.dtx`. Note: At the time of writing, this has not yet been uploaded to CTAN.

[4] Lars Hellström: *The xdoc package — experimental reimplementations of features from doc, second prototype*, 2000–2003; CTAN: `macros/latex/contrib/xdoc/xdoc2.dtx`.

[5] Roger Kehr: *xindy — A Flexible Indexing System*; CTAN: `indexing/xindy/`.

[6] The LaTeX3 Project: *The LaTeX Project Home Page*; HTTP://`www.latex-project.org/`.

[7] Frank Mittelbach: *An environment for multicolumn output*; CTAN: `macros/latex/required/tools/multicol.dtx`.

[8] Frank Mittelbach, B. Hamilton Kelly, Andrew Mills, Dave Love, and Joachim Schrod: *The doc and shortvrb Packages*, The LaTeX3 Project; CTAN: `macros/latex/base/doc.dtx`.

Change History

v 0.03

`docindex/changes` instance: Added `\@nobreakfalse` in docidx2e implementation; the first item in the index does not directly follow a `\section`-type heading. (LH) . . . 27

`docindex/std` template: Added *parskip* keyval. (LH) . . . 23

`indexitem/aloneaccept` template: `\protected@edef`ing the macros `\DI@item@join` and `\DI@item@nojoin`. (LH) . . . 20
  Added the *nojoin-extra* key. (LH) . . . 20

`indexitem/fixed` template: `\protected@edef`ing the macros `\DI@item@join` and `\DI@item@nojoin`. (LH) . . . 16
  Added the *nojoin-extra* key. (LH) . . . 16
  Added `\nopagebreak` from `\efill` to default for *nojoin-extra* key. (LH) . . . 16

`usedocindexps` option: Added `\@nobreakfalse` in docidx2e implementation; the first item in the index does not directly follow a `\section`-type heading. (LH) . . . 25

v 1.00

General: The index file is no longer a `thedocindex` environment—the layout must instead be set by the command which `\input`s the index. Introduced the `\docindexguard` command to handle situations with incompatible index styles. (LH) . . . 11
  Using `single` template rather

than the `std` template for the `indexitem`*n* instances of type `justification`. (LH) 14

`docindex/changes` instance: Added `\makeatletter` in `docidx2e` implementation; it doesn't hurt and it is sometimes necessary (when commands that expand to private control sequences are used in the argument of `\changes`). (LH) 27

`docindex/std` template: Added *file-name* and *file-extension* keys, removed the *default-prologue* and *default-epilogue* keys. (LH) 22

Made *reserved-height* work even when *columns* is 1 by using the multicol macro `\enough@room`. (LH) 23

Added *page-compositor* keyval. (LH) 23

The *-font* keyvals renamed to *-setup*, but the type stayed the same (f0). (LH) 22

`indexitem/aloneaccept` template: Changed condition for accepting a join from "neighbouring item levels are lower" to "neighbouring item levels are not equal to". (LH) 19

`\indexnewletter`: Made it `\outer`. (LH) 9

`\XD@page@compositor`: Changed it from `\PageCompositor-` to `\+`. (LH) 10

Index

Numbers written in italic refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; numbers in roman refer to the code lines where the entry is used.

Symbols

`\+` 71, 491, 513, 564, 604, 650

B

`\DI@body@setup` 472, 504, 526

C

`\c@GlossaryColumns` 652, 653, 665
`\c@IndexColumns` 566, 567, 580, 606, 607, 618
`\DI@columns` . . . 463, 501, 517, 518, 533
`\columnsep` . . . 466, 502, 556, 596, 646

D

`docindex` template type 439
  `changes` instance 630
  `index` instance 543
  `std` template 441
    body-setup keyval (f0) *23*
    column-sep keyval (l) *23*
    columns keyval (C) *23*
    epilogue-setup keyval (f0) *23*
    file-extension keyval (n) *22*
    file-name keyval (n) *22*
    item1 keyval (i) *23*
    item2 keyval (i) *23*
    item3 keyval (i) *23*
    letter-format keyval (f1) *23*
    letter-skip keyval (L) *23*
    page-compositor keyval (f0) . . *23*
    pagestyle keyval (n) *23*
    parskip keyval (l) *23*
    prologue-setup keyval (f0) *23*
    reserved-height keyval (L) *23*
`\docindexguard` 83, 98

E

`\enough@room` 521, 536, 569, 583, 609, 621, 655, 668
`\DI@epilogue@setup` 475, 505, 539

F

`\DI@file@ext` . . 448, 496, 516, 528, 540
`\DI@file@name` . 445, 495, 516, 528, 540

G

`\glossary@prologue` 681
`\GlossaryMin` 653, 655

H

`\DI@hasfig` 184, 203, 230, 347, 364, 389

I

`\ifDI@postjoin@` . . . 204, 223, 365, 382
`\ifDI@prejoin@` 201, 214
`\IfFileExists` 516, 565, 605, 651

- `\DI@ind@setup` . . 83, 527, 576, 615, 662
- `\index@prologue` 677
- `\indexitem` *8*, 20, 27
- `indexitem` template type 169
  - `aloneaccept` template 336
    - *fig-action* keyval (f2) *19*
    - *join-extra* keyval (f0) *20*
    - *justification-setup* keyval (i) . . *19*
    - *nofig-action* keyval (f1) *19*
    - *nojoin-extra* keyval (f0) *20*
    - *offjoin-extra* keyval (f0) *20*
    - *ownlevel* keyval (C) *19*
    - *post-join* keyval (b) *20*
  - `aloneaccept2` instance 395
  - `aloneaceept3` instance 395
  - `fixed` template 173
    - *fig-action* keyval (f2) *15*
    - *join-extra* keyval (f0) *16*
    - *justification-setup* keyval (i) . . *15*
    - *nofig-action* keyval (f1) *15*
    - *nojoin-extra* keyval (f0) *16*
    - *offjoin-extra* keyval (f0) *16*
    - *post-join* keyval (b) *16*
    - *pre-join* keyval (b) *15*
  - `fixed-a3r` instance 283
  - `fixed-r1a` instance 283
  - `fixed-r2a-nocomma` instance . . . 283
  - `fixed1` instance 235
  - `fixed2` instance 235
  - `fixed3` instance 235
- `\DI@indexitem` 27
- `\DI@indexitem@` 27
- `\DI@indexitem@`⟨*level*⟩ *9*
- `\DI@indexitem@i` 451, 497, 553, 593, 643
- `\DI@indexitem@ii` 454, 498, 554, 594, 644
- `\DI@indexitem@iii` 457, 499, 555, 595, 645
- `\IndexMin` 567, 569, 607, 609
- `\indexnewletter` 48, 49, 98
- `\input` 528, 577, 616, 663
- `\DI@item@join` 61, 169, 215, 219, 224, 252, 265, 278, 299, 303, 314, 318, 325, 331, 376, 378, 383, 409, 416, 427, 434
- `\DI@item@justification` . 175, 215, 217, 244, 246, 257, 259, 270, 272, 294, 296, 309, 311, 324, 325, 338, 372, 374, 376, 403, 405, 407, 409, 421, 423, 425, 427
- `\DI@item@nojoin` 57, 62, 169, 217, 220, 225, 245, 253, 258, 266, 271, 279, 295, 300, 304, 310, 315, 319, 332, 372, 374, 379, 384, 405, 407, 417, 423, 425, 435, 529, 578, 617, 664

J

- `\DI@join@extra` 194, 208, 224, 356, 367, 383
- `justification` template type 134
  - `indexitem1` instance 135
  - `indexitem2` instance 135
  - `indexitem3` instance 135
- `\justification@g` 189, 196, 206, 210, 352, 366

L

- `\DI@last@level` 27, 60
- `\@letinstance` 114, 126, 244, 257, 270, 294, 309, 324, 403, 421, 497–499, 553–555, 593–595, 643–645
- `\DI@letter@format` 59, 482, 507, 559, 599, 649
- `\DI@letter@skip` 58, 478, 506, 558, 598, 648

M

- `\makeatletter` 660
- `\multicolsep` 523, 537, 571, 584, 611, 622, 657, 669

N

- `\DI@nofig` . 181, 202, 222, 344, 363, 381
- `\DI@nojoin@extra` 191, 205, 227, 353, 358, 366, 368, 386

O

- `\DI@offjoin@extra` 198, 209, 225, 359, 368, 384
- `oldkeywords` option *4*, 9, 74
- options:
  - `oldkeywords` *4*, 9, 74
  - `usedocindexps` 10, 543

P

- `\DI@pagestyle` . 485, 512, 525, 530, 531
- `\pfill` 183, 203, 250, 263, 276, 302, 317, 329, 346, 364, 414, 432
- `\postmulticols` 536, 583, 621, 668
- `\PrintChanges` 676
- `\PrintIndex` 676
- `\DI@prologue@setup` . 469, 503, 519, 522

R

- `\DI@reserved@height` 460, 500, 519, 521

T

- `\@tempa` 126, 127, 130

\DI@this@level 341, 362, 371, 373
\TP@I{⟨*collection*⟩}{⟨*type*⟩}{⟨*name*⟩} *12*
\TP@T{⟨*type*⟩} *13*
\typeout 540, 587, 625, 672

U

\UseCollection 107
usedocindexps option 10, 543
\UseInstance . . 125, 480, 508, 677, 681

X

\XD@glossary@keyword 74
\XD@index@keyword 74
\XD@page@compositor 74