The package imakeidx*

Claudio Beccari† Enrico Gregorio‡

Contents

1 Introduction 1
2 Package usage 2
3 Specific package commands 3
4 If something goes wrong 7
5 Hints 9
6 Implementation 10

Abstract

This package exploits the `\write18` facility of modern TEX system distributions that allows to run system commands while typesetting a document written with the LATEX mark up. By so doing, the index or indices, that are usually typeset at the very end of the document, are possibly split and sorted so as to include them in the document itself. This process has some minor limitations: it's impossible to start an index before all other pages have been ejected and to have the automatic run of the index sorting program.

1 Introduction

It's been some years now that the typesetting engine of the TEX system is just *pdftex*; the original Knuthian *tex* is still corrected by D. E. Knuth himself, but is frozen, according to his will; it is still distributed by every TEX distribution, but in practice only *pdftex*, *xetex* or *luatex* are used as the interpreter of every macro package and the true typesetter engine.

This program *pdftex* was originally born with the facility of producing either a pdf output file, as its name suggests, or a dvi file. Since then it has been enriched with many upgrades, also with regard to the evolution of the PDF language itself. It also incorporates the extensions of ε-TEX and has the ability to open a shell so as to call system commands with their arguments. The same is true for *xetex* and *luatex*.

This facility, since the TEX Live 2010 distribution, is official, but is sort of restricted, in the sense that the TEX system configuration file contains a list of

*Version number v1.1a; last revision 2012/09/07.

†`claudio dot beccari at gmail dot com`
‡`Enrico dot Gregorio at univr dot it`

"safe" system commands that can be run by *pdftex*; presently the only program relevant for this package is *makeindex*. This precaution is necessary in order to avoid running malicious code. Other programs can be run, though, but it's necessary to expressly tell *pdftex* that it can do so; this authorisation is given by means of a suitable program option, as explained below.

This package will exploit this facility in order to run a perl script that is capable of splitting a raw index file into different chunks and to run the *makeindex* TEX system program so as to sort and format the index entries according to a specified index style file. Once the shell is terminated, the *pdftex* program resumes its work and possibly prints the various formatted indices produced in previous step. In this way the document indices are always synchronous with their document and no further *pdftex* runs are necessary.

In order to reach this goal, it necessary to enable the *pdftex* engine to run the so-called `\write18` facility; depending on the distribution and the shell editor that is being used to work on a specific document, it is necessary to add `-shell-escape` (or `--enable-write18` for MiKTEX) to the command with which *pdftex* is launched, possibly by the shell editor. What's been said for *pdftex* goes without change for the *xetex* and *luatex* typesetting engines.

If LuaLATEX is used and *luatex* is version 0.42 to 0.66, it's impossible to distinguish whether the restricted shell escape is active or not, so the automatic procedure will be tried anyway, unless disabled with the `noautomatic` package option. With version 0.68 or later, the behavior is the same as with the other engines.

Note

The first public version of this package was not compatible with the memoir class. Since version 1.1 it is; however, one has to keep in mind that all index processing is done with the methods of the present package, and *not* with memoir's; however the syntax used is the same and there should be no problem.

2 Package usage

This package is invoked as usual by means of a `\usepackage` command:

`\usepackage[`⟨*options*⟩`]{imakeidx}`

The available ⟨*options*⟩ consist in a comma separated list of the following options:

`makeindex` in order to use the *makeindex* sorting an formatting engine; this option is the default and is mutually exclusive with the next option.

`xindy` in order to use the *xindy* sorting and formatting engine; `texindy` is an alias for `xindy` and actually it's the script *texindy* which is called by this package.

`noautomatic` disables the automatic splitting and running of the system programs; this option might be used to save time when one knows for sure that

the index files are already OK and do not need to be refreshed. Actually the time spent in splitting, sorting and formatting is so short that this option might be useful only when very lengthy indices are being processed.

`nonewpage` inhibits the new page command to be issued when using an article type document class and multiple indices are being typeset. We don't see why someone would use multiple indices in an article (except possibly for package documentations, which usually provide a macro index and a list of changes).

`quiet` suppresses all messages about manual index processing.

`original` uses the class provided `theindex` environment for typesetting the indices; it is implicitly set if the document class option `twocolumn` has been specified.

`splitindex` calls the *splitindex* script by Markus Kohm, which is included in every TEX Live distribution since 2009. With this option all index entries, which are written in one raw file, are successively split into all the requested index files; in this way there is virtually no limit on the number of indices that is possible to create for a particular document.

The last described option deserves an explanation. LaTeX can write on a limited number of files during a run, and some of these *output streams* are already reserved (among these: aux file, table of contents, list of figures, list of tables). When more than one index is produced, there's the risk to run off the number of writable files, because normally imakeidx reserves an output stream for each index. So the `splitindex` option comes to rescue: with it only *one* index file is written. At the first `\printindex` command, the program *splitindex* is called; it splits the large index file into as many parts as the number of requested indices; after this, *makeindex* (or *xindy*) can do its job. In this way only one output stream is needed during the LaTeX run.

When should you apply this option, then? With one index it's useless, you should begin to consider it for two or more indices and definitely use it if you get the error

```
! No room for a new \write
```

Apart from this case, with or without it the results are the same. See section 4 to see what files are written during the LaTeX run with or without the option.

3 Specific package commands

As it is customary when just one index is produced, the standard LaTeX facilities, i.e. the commands `\makeindex`, `\index`, and `\printindex` must be used. This package redefines them so as to produce multiple indices and defines some others. The first three of the following commands may be used only in the preamble.

`\makeindex` with the syntax:

`\makeindex[`⟨*key-values*⟩`]`

where ⟨*key-values*⟩ is a comma separated list of key-value assignments of the form: `key=value`; the available keys are the following:

name is the symbolic name for an index; if this key is not specified, it defaults to the value of the `\jobname` control sequence, in other words the name of the current main `.tex` file, i.e., the file that `\input`s and/or `\include`s all the files of the complete document. This symbolic name is necessary only when doing multiple indices and is used with the `\index` command to point to the right index.
Example: `name=nameidx`

title is the title that is typeset at the beginning of the specific index; if not specified, the `\indexname` value is used.
Example: `title=Index of names`

program is the name of the system program that is used to sort and format an index; valid choices are *makeindex*, *xindy*, or *texindy*. If not specified the program specified among the package options is used. If no option is specified, *makeindex* is used. In order to use *xindy*, it's necessary to call (*pdf*)*latex* with the shell escape command line option.
Example: `program=xindy`

options is the list of options to be passed to the sorting and formatting program; this list is a balanced text of program options, separated with the syntax required by the sorting and formatting program. For example, in order to use a different *makeindex* sorting and formatting style mystyle.ist and avoiding any message in the screen output write `options=-s mystyle`

noautomatic is a boolean key that defaults to `false`; you can set it to `true` by simply listing its key in the key-value list, without necessarily specifying the `=true` part. If specified the index sorting program won't be called during the *pdftex* run for this particular index.

intoc is a boolean variable that defaults to `false`; if you want to set it `true` you must simply list this key in the key-value list, with no need of specifying the `=true` part. By setting this key to `true` an entry for this particular index is put in the table of contents.

columns accepts an integer representing the number of columns in the index; this is silently ignored if the `original` or the `twocolumn` options are set; the number can even be 1.
Example: `columns=3`

columnsep accepts a dimension representing the separation between index columns; the default is 35 pt as in the standard classes.
Example: `columnsep=15pt`

columnseprule is boolean; if it is set, a rule will appear between the index columns.

`\indexsetup` with the syntax:

`\indexsetup[`⟨*key-values*⟩`]`

where again ⟨*key-values*⟩ is a comma separated list of key-value assignments; the available keys are:

level which takes as value a sectioning command such as `\chapter` or `\chapter*`. Actually any command with an argument will do and will receive the index title as its argument. The default is `\chapter*` or, if the class doesn't provide chapters, `\section*`.

toclevel which takes as value a sectioning command *name* such as `section` to indicate the level at which we want the indices appear in the table of contents.

noclearpage is a boolean option; when set, no `\clearpage` will be issued between indices. You might want to set it in order to have a 'chapter of indices'; in this case you are responsible for setting the right value of the above keys. For example

```
\indexsetup{level=\section*,toclevel=section,noclearpage}
...
\chapter*{Indices}
\printindex
\printindex[names]
\printindex[objects]
```

firstpagestyle which takes as value a page style, default `plain`. You might want to set it to `empty` or some other page style defined by the class or by yourselves.

headers which takes as value the left and right marks. You might want to use this for disabling automatic uppercasing, by saying `headers={\indexname}{\indexname}`; notice that this value should always be a pair of braced texts.

othercode which takes as value arbitrary TEX code that will be executed at the beginning of index entries typesetting. For example you might want to change here the setting of `\parskip`.

`\splitindexoptions` must have as its argument the command line option to *splitindex*; this might be necessary on some systems. The default is `-m ""`, because we want it only for splitting the large index file into its components which are later processed by this package.

`\index` with the syntax:

`\index[`⟨*name*⟩`]{`⟨*entry*⟩`}`

inserts ⟨*entry*⟩ into the raw index file; upon splitting it in different files, this particular entry is listed in the specific index file with name ⟨*name*⟩; if no name is specified, this ⟨*entry*⟩ is added to the default index with name

`\jobname`. The ⟨*entry*⟩ should be written according to the particular syntax of the sorting and formatting program.

`\indexprologue` with the syntax:

`\indexprologue[`⟨*spacing*⟩`]{`⟨*text*⟩`}`

is used to define some text to go between the index header and the entries; the ⟨*spacing*⟩ should be a vertical space command such as `\vspace{36pt}` (default is `\bigskip`), controlling the spacing between the prologue and the index proper. The command affects only the next index produced by `\printindex` and is best placed just before this command.

`\printindex` with the syntax:

`\printindex[`⟨*name*⟩`]`

is used to typeset the particular index named ⟨*name*⟩; if no optional argument is specified, the default index with name `\jobname.ind` is typeset. Actually this command activates all the mechanism of closing the output to the raw index file, shelling out, possibly calling the *splitindex* script in order to divide the single raw file generated by (*pdf*)*latex* into distinct raw files according to the default or specified ⟨*name*⟩s for each index, calling the sorting and formatting program on each of these split raw files (unless inhibited by a *noautomatic* option; in which case a warning is issued in order to remember the typesetter that this particular index has not been processed), producing the sorted and formatted `.ind` files, and eventually inputs and typesets these formatted files. Deep breath.

Let's see an example. The sequence of commands

```
...
\usepackage{imakeidx}
...
\makeindex[title=Concept index]
\makeindex[name=persons,title=Index of names,columns=3]
...
\begin{document}
...
And this is the end of the story.

\printindex

\indexprologue{\small In this index you'll find only
  famous people's names}
\printindex[persons]
\end{document}
```

will produce two indices. Entries for the first one must be typed as `\index{gnu}`, while entries for the second are of the form `\index[persons]{Lamport, Leslie}`. The prologue will be printed (full line) only in the "Index of names", which will be typeset in three columns.

When the `original` option is set, maybe implicitly because of `twocolumn`, `\indexsetup` and the keys *columns*, *columnsep* and *columnseprule* for `\makeindex` have no effect.

4 If something goes wrong

Since imakeidx relies on good cooperation between package options and command line options for the LaTeX run, in some cases it may happen that the indices are not correctly built or built at all.

If you use only *makeindex* and TeX Live 2010 or later, then you shouldn't need anything special, since *makeindex* is among the safe programs allowed to be called during a LaTeX run, be it *latex*, *pdflatex*, *xelatex*, or *lualatex*. When the options `splitindex`, `xindy` or `texindy` are specified (globally or locally), the LaTeX run should be called with `-shell-escape` (which is `-enable-write18` for MiKTeX) or the `noautomatic` option should be specified when loading imakeidx.

Let's look at a couple of examples. In both we suppose that the document `mybook.tex` defines two indices through

```
\makeindex[...]
\makeindex[name=secondary,...]
```

where `...` denotes possible options excluding `name`.

First of all we examine the case when imakeidx is called *without* `splitindex`. Two files called `mybook.idx` and `secondary.idx` will be written during the LaTeX run. At the corresponding `\printindex` command, *makeindex* will act on each of them producing the files `mybook.ind`, `mybook.ilg`, `secondary.ind` and `secondary.ilg`. The `.ind` files contain the relevant `theindex` environment with alphabetized entries, while in the `.ilg` files *makeindex* will write its log. You can check in `mybook.log` whether the *makeindex* run has been executed by searching for a line

```
runsystem(makeindex <...>)...executed
```

where `<...>` stands for the rest of the command line in the particular case. If this line is not present, then *makeindex* has not been called; this happens when you didn't specify the shell escape command line option for the LaTeX run or the restricted shell escape is not active; also, of course, if you set the `noautomatic` option for the index.

When using *splitindex*, the situation is different. During the LaTeX run, only a large index file called `mybook.idx` file gets written; the first `\printindex` command will call *splitindex* (shell escape *must* be active), which will produce the two partial index files `mybook-mybook.idx` and `mybook-secondary.idx`. These two files will be processed by *makeindex* producing the four files `mybook-mybook.ind`, `mybook-mybook.ilg`, `mybook-secondary.ind` and `mybook-secondary.ilg`. The line

```
runsystem(splitindex <...>)...executed
```

	\makeindex \makeindex[name=secondary]	
	without `splitindex`	with `splitindex`
(at `\begin{document}`)		
	`mybook.idx` `secondary.idx`	`mybook.idx`
(at `\printindex`)		

	`mybook.ind`
	`mybook.ilg`
	`secondary.ind`
	`secondary.ilg`
		`mybook-mybook.idx`
		`mybook-secondary.idx`
		`mybook-mybook.ind`
		`mybook-mybook.ilg`
		`mybook-secondary.ind`
		`mybook-secondary.ilg`

Table 1: Files written during a LaTeX run

in `mybook.log` will tell that the splitting has been done (see later on if this doesn't seem true). In table 1 you can see what files are produced when the first two lines are in the preamble.

Everything is the same when using *texindy* for alphabetizing, except that, by default, it doesn't write `.ilg` files. If you want them, add `options=-t`⟨*name*⟩`.ilg` to the relevant `\makeindex` command, in our example it should be

```
\makeindex[...,options=-t mybook.ilg]
\makeindex[name=secondary,...,options=-t secondary.ilg]
```

The name of the `.ilg` file *must* be specified. Remember, though, that *xindy* `.ilg` files may turn out to be very large.

When something different from expected appears to take place, check also the time stamps of the produced files; if they are older than `mybook.log`, it means that they have not been written in the last run. The most common case is that you forgot to activate the shell escape feature (which is not necessary with TeX Live 2010 or later, provided you use only *makeindex*).

Another cause of malfunction might be a wrong option passed to *makeindex*, *texindy* or *splitindex*. For example, if you specify a style option for *makeindex* such as `options=-s mystyle.ist` and the style file is missing or its name is mistyped, the run of *makeindex* will result in `mybook.log`, but it will be aborted and the TeX program has no control over this process. In this case the `.ilg` and `.ind` files will not be produced and you can spot the problem by checking the time stamps. On some systems a message such as

```
Index file mystyle.ist not found
Usage: makeindex [-ilqrcgLT] [-s sty] [-o ind] [-t log] [-p num]
```

may appear on the screen, but often this window gets closed before you realize you have a problem. The time stamp is the best clue to detect such problems.

Shell hackers may be able to redirect the `stderr` stream to a file, but this requires skills that can't be explained here, because they require tens of different tricks, depending on what method is used to start a LaTeX run. From the command line, assuming *bash*, it would be something like

```
pdflatex -shell-escape mybook.tex 2>latex-errors
```

If shell hackers know a way to access the exit status of the called program, we'd be glad to implement a supplementary check.

5 Hints

Actually this package reaches two goals: (*a*) it typesets the indices of a specific document in just one run, and (*b*) it lets the author/typesetter produce documents with multiple indices.

If you redefine yourself the `theindex` environment, please remember not to number the chapter or section that introduces the index if you ask for the *intoc* option; either use the commands `\chapter*` or the `\section*` respectively and the *intoc* option or don't use this option and redefine your `theindex` environment with numbered chapter or section commands, that will put the index titles directly into the table of contents. You may use the idxlayout package by Thomas Titz, which offers many functions for index typesetting customization and is compatible with our package; remember to load idxlayout after imakeidx. This package has a similar function to our `\indexprologue`, called `\setindexprenote`; however idxlayout doesn't reset the index prologue, which must be declared anew or disabled with `\noindexprenote` before the next `\printindex` command.

If by chance you get double entries into the table of contents, eliminate the *intoc* option from your calls; your class and packages are already taking care of it. The package tocbibind should be loaded with the `noindex` option, otherwise it would interfere with our redefinition of `theindex`.

If you redefine your `theindex` environment by means of other packages, pay attention that these redefine a real `theindex` environment with this very name; if they create an environment with a different name, imakeidx can't take care of the indices production (in particular the TeX system program *makeindex* creates a sorted and formatted `.ind` file that refers explicitly to the `theindex` environment), and it can't take care of the table of contents entry and of the position of the hyper link anchor needed to navigate your document by means of hyper links.

Use freely the options and the key values in order to reach the desired results, but you are advised to prepare in advance the styles for composing the various indices in a proper way; for example, if you use a titled style for the index, where the index sections are distinguished with a bold face title or alphabetic letter, you have to set up a `.ist` file, such as `myindexstyle.ist`, made up like this:

```
headings_flag 1
heading_prefix   "\\par\\penalty-50\\textbf{"
```

```
heading_suffix   "}\\\\\*\~\\\\\*"
symhead_positive "Symbols"
symhead_negative "symbols"
numhead_positive "Numbers"
numhead_negative "numbers"
delim_0 ",\~"
```

where the numeric and non alphabetic entries have different titles. But, say, you are making also an index where the entries are file names, and for some names only the extension is entered; the extensions start with a dot, so the sorting program will sort these names at the beginning of the sorted index file, but you won't like to have a title such as "Symbols"; you probably prefer to have a title such as "Extensions"; therefore you have to prepare a different index style file, such as this one:

```
headings_flag 1
heading_prefix   "\\par\\penalty-50\\textbf{"
heading_suffix   "}\\\\\*\~\\\\\*"
symhead_positive "Extensions"
symhead_negative "extensions"
numhead_positive "Numbers"
numhead_negative "numbers"
delim_0 ",\~"
```

This done, besides requiring the use of this package, you have to declare the `\makeindex` command with the necessary options; pay a particular attention to the options that involve the index symbolic name, the index title, the index style, the fact that the index titles shall appear in the table of contents, and if you are preparing an e-book, you probably would like to hyper link both the page numbers and the index titles to the proper locations. *pdflatex* will do everything for you but be careful not to confuse it with illogical index entries.

Especially with multiple indices it is important that you are consistent in putting the right information in the right index and with a consistent mark-up. Define yourself appropriate macros so that, for example, personal names are consistently typeset, say, in caps and small caps and are entered into a specific index; you may even create one command to typeset the name in the document and replicate the same name in the index.

Of course there is no program that can decide at your place what and where to index each piece of information; this is a task for humans. Soooooo...

HAPPY TEXING!

6 Implementation

The heading to the file is in common with the documentation file, and has already been taken care of. But we require the xkeyval package, in order to handle the key-value lists.

Notice that in order to create a specific name space so as to avoid possible conflicts with other packages, all the commands defined in this package are prefixed with the string `imki@`.

```
\RequirePackage{xkeyval}
```

We define the various options and their defaults. After `\ProcessOptions`, we set anyway the `original` option if the document class has been given the `twocolumn` option, which is incompatible with multicol. We define also an internal alias for `\immediate\write18`, a rudimentary check for the typesetting engine and a macro for modifying the command line call to *splitindex*.

```
\DeclareOption{xindy}{\def\imki@progdefault{texindy}}
\DeclareOption{texindy}{\def\imki@progdefault{texindy}}
\DeclareOption{makeindex}{\def\imki@progdefault{makeindex}}
\newif\ifimki@disableautomatic
\DeclareOption{noautomatic}{\imki@disableautomatictrue}
\newif\ifimki@nonewpage
\DeclareOption{nonewpage}{%
  \imki@nonewpagetrue\imki@disableautomatictrue
}
\newif\ifimki@splitindex
\DeclareOption{splitindex}{\imki@splitindextrue}
\newif\ifimki@original
\DeclareOption{original}{\imki@originaltrue}
\DeclareOption{quiet}{\AtEndOfPackage{%
  \let\imki@finalmessage\@gobble
  \let\imki@splitindexmessage\relax}}
\ExecuteOptions{makeindex}
\ProcessOptions\relax

\if@twocolumn\imki@originaltrue\fi
\def\imki@exec{\immediate\write18}
\def\imki@engine{(pdf)latex}
\RequirePackage{ifxetex,ifluatex}
\ifxetex\def\imki@engine{xelatex}\fi
\ifluatex % luatex doesn't have \(pdf)shellescape
 \def\imki@engine{lualatex}
  \ifnum\luatexversion<68
    \chardef\imki@shellescape\@ne % no way to know the value
  \else
    \RequirePackage{pdftexcmds} % provides \pdf@shellescape
    \chardef\imki@shellescape\pdf@shellescape
  \fi
  \let\imki@exec\pdf@system
\fi
\edef\imki@splitindexoptions{-m \string"\string"}
\def\splitindexoptions#1{\g@addto@macro\imki@splitindexoptions{ #1}}
\@onlypreamble\splitindexoptions
```

While experimenting we found out that some classes or packages are either incompatible with this one, or must be faked in order to pretend they have been

loaded.

There is a serious incompatibility with the memoir class. In facts memoir puts all index entries in the main `.aux` file and extracts them to the various raw index files at `\end{document}` time. This means that no raw index file output stream has been defined, and therefore this package can't close it; moreover it can't typeset the indices before `\end{document}` because they are not yet available. Therefore if memoir is the active class, we will hijack its index mechanism replacing it with ours.

On the opposite we pretend that package makeidx or package multind have been loaded, so that hyperref can play with their commands, that are substantially the same as those used here. By so doing those packages are inhibited from being loaded after this one.

```
\@namedef{ver@makeidx.sty}{3000/12/31}
\@ifpackageloaded{multind}
  {\PackageError{imakeidx}{Incompatible package 'multind' loaded}
    {This package is incompatible with multind, don't load both.%
     \MessageBreak\@ehc}}
  {\@namedef{ver@multind.sty}{3000/12/31}}
```

At the same time we redefine some commands defined by makeidx and we define the default English names for the `\see` and `\seealso` commands. We use `\providecommand` so that, if makeidx has already been loaded, we do not redefine things that have already been defined.

```
\providecommand*\see[2]{\emph{\seename} #1}
\providecommand*\seealso[2]{\emph{\alsoname} #1}
\providecommand*\seename{see}
\providecommand*\alsoname{see also}
```

From here on, some commands are duplicated; this depends on the fact that the behavior must be different when using *splitindex* or not. The memory occupied by the useless commands will be cleared at the end of package.

```
\providecommand*\makeindex{} % to use \renewcommand safely
\renewcommand{\makeindex}[1][]{\imki@makeindex{#1}}
% \@onlypreamble\makeindex % Already in latex.ltx
```

This package implementation of `\makeindex` sets default values for the keys, then evaluates its argument (which is the optional argument to `\makeindex`) and calls two other macros. After that we have to reset the defaults.

```
\def\imki@makeindex#1{%
  \def\imki@name{\jobname}%
  \def\imki@title{\indexname}%
  \edef\imki@program{\imki@progdefault}%
  \let\imki@options\space
  \KV@imki@noautomaticfalse\KV@imki@intocfalse
  \setkeys{imki}{#1}%
  \ifimki@splitindex\KV@imki@noautomaticfalse\fi
  \imki@build\imki@name
  \imki@startidx\imki@name
  \imki@resetdefaults
```

```
}
```

Here are the keys. As usual, the `imki@` prefix is used to distinguish anything that is being defined in this package, even the keys.

```
\define@key{imki}{name}{\def\imki@name{#1}}
\define@key{imki}{title}{\def\imki@title{#1}}
\define@choicekey{imki}{program}[\imki@val\imki@nr]
  {makeindex,xindy,texindy}{%
    \ifcase\imki@nr\relax
      \def\imki@program{makeindex}%
    \or
      \def\imki@program{texindy}%
    \or
      \def\imki@program{texindy}%
    \fi}
\define@key{imki}{options}{\def\imki@options{ #1 }}
\define@boolkey{imki}{noautomatic}[true]{}
\define@boolkey{imki}{intoc}[true]{}
\define@key{imki}{columns}{\def\imki@columns{#1}}
\define@key{imki}{columnsep}{\def\imki@columnsep{#1}}
\define@boolkey{imki}{columnseprule}[true]{}
\def\imki@resetdefaults{%
  \def\imki@options{ }%
  \def\imki@columns{2}\def\imki@columnsep{35\p@}%
  \KV@imki@columnseprulefalse
  \KV@imki@intocfalse\KV@imki@noautomaticfalse}
\imki@resetdefaults
```

The control sequence `\imki@build` defines a control sequence to hold the setup for an index to be used when the index is sorted and printed

```
\def\imki@build#1{%
  \toks@{}%
  \imki@dokey\imki@title
  \imki@dokey\imki@program
  \imki@dokey\imki@options
  \imki@dokey\imki@columns
  \imki@dokey\imki@columnsep
  \ifKV@imki@noautomatic
    \addto@hook\toks@{\KV@imki@noautomatictrue}%
  \else
    \addto@hook\toks@{\KV@imki@noautomaticfalse}%
  \fi
  \ifKV@imki@intoc
    \addto@hook\toks@{\KV@imki@intoctrue}%
  \else
    \addto@hook\toks@{\KV@imki@intocfalse}%
  \fi
  \ifKV@imki@columnseprule
    \addto@hook\toks@{\KV@imki@columnsepruletrue}%
  \else
```

```
    \addto@hook\toks@{\KV@imki@columnseprulefalse}%
  \fi
  \expandafter\edef\csname imki@set@#1\endcsname{\the\toks@}%
}
```

Comand `\imki@dokey` receives as argument the text of the values assigned to certain keys, and adds them to the options token list.

```
\def\imki@dokey#1{%
  \expandafter\addto@hook\expandafter\toks@\expandafter{%
    \expandafter\def\expandafter#1\expandafter{#1}}}
```

Command `\imki@startidx` defines the output stream(s); the macro with suffix `split` is used when *splitindex* is not enabled, the one with suffix `unique` is used otherwise. In the case of many indices, the symbolic name for an index named 'pippo' is `\pippo@idxfile` corresponding to the file `pippo.idx`. When *splitindex* is enabled, the only output stream is called `\@indexfile` as in standard LaTeX, corresponding to `\jobname.idx`.

```
\def\imki@startidxsplit#1{%
  \if@filesw
    \def\index{\@bsphack
      \@ifnextchar [{\@index}{\@index[\jobname]}}
    \expandafter\newwrite\csname #1@idxfile\endcsname
    \immediate\openout \csname #1@idxfile\endcsname #1.idx\relax
    \typeout{Writing index file #1.idx}%
  \fi}
```

We define a switch which is set to true when a `\makeindex` command is given: with *splitindex* we open only one stream.

```
\newif\ifimki@startedidx
\def\imki@startidxunique#1{%
  \if@filesw
    \ifimki@startedidx\else
      \newwrite\@indexfile
      \immediate\openout\@indexfile\jobname.idx%
      \global\imki@startedidxtrue
    \fi
    \def\index{\@bsphack
      \@ifnextchar [{\@index}{\@index[\jobname]}}
    \expandafter\let\csname #1@idxfile\endcsname\@empty
    \typeout{Started index file #1}%
  \fi}
```

Provide a default definition for `\index`; when a `\makeindex` command is given and LaTeX is writing on auxiliary files, `\index` will be redefined, as seen before. When index files are written, `\index` always calls `\@index`. Some code is borrowed from memoir.cls, but heavily modified. We want `\@wrindex` to be defined with two arguments, so that hyperref can hook into it just like it does with the similar commands defined by the old packages multind and index.

```
\renewcommand{\index}[2][]{\@bsphack\@esphack}
\def\@index[#1]{%
```

```
  \@ifundefined{#1@idxfile}%
  {\PackageWarning{imakeidx}{Undefined index file `#1'}%
   \begingroup
   \@sanitize
   \imki@nowrindex}%
  {\edef\@idxfile{#1}%
   \begingroup
   \@sanitize
   \@wrindex\@idxfile}}
\def\imki@nowrindex#1{\endgroup\@esphack}
```

Command `\@wrindex` must be duplicated; we have to call it the same as usual in order to support hyperref. But the real name will be given at the end.

```
\def\imki@wrindexsplit#1#2{%
  \expandafter\protected@write\csname#1@idxfile\endcsname{}%
    {\string\indexentry{#2}{\thepage}}%
  \endgroup
  \@esphack}
\def\imki@wrindexunique#1#2{%
  \protected@write\@indexfile{}%
    {\string\indexentry[#1]{#2}{\thepage}}%
  \endgroup
  \@esphack}
```

Compilation of the indices is disabled if `-shell-escape` has not been given or the restricted mode is not active; in this case we emit a warning. XeTeX has `\shellescape` instead of `\pdfshellescape`, so we take care of this (hoping that users or packages don't define a `\shellescape` command). In any case we define an internal version of this command. In the case of *luatex* we can't emit the proper messages if *luatex* is not version 0.68 or later. The conditional `\ifKV@imki@noautomatic` is defined by `\define@boolkey` above.

```
\def\imki@shellwarn{}
\ifdefined\imki@shellescape % luatex
\else
  \@ifundefined{shellescape}
    {\let\imki@shellescape\pdfshellescape} % pdftex
    {\let\imki@shellescape\shellescape} % xetex
\fi
\ifnum\imki@shellescape=\z@
  \let\KV@imki@noautomaticfalse\KV@imki@noautomatictrue
  \KV@imki@noautomatictrue
  \def\imki@shellwarn{\MessageBreak or call \imki@engine\space with
    -shell-escape}
\fi
```

Do the same if *noautomatic* has been given as an option.

```
\ifimki@disableautomatic
  \let\KV@imki@noautomaticfalse\KV@imki@noautomatictrue
  \KV@imki@noautomatictrue
\fi
```

Now we set up the `theindex` environment. If the `original` option is set, we simply patch the class definition in order to call the macro that does the work related to the table of contents. Otherwise we define a new `theindex` environment, based on the standard, but using, if the number of columns is greater than one, the `multicols` environment. Users needing a different setup can use the `\indexsetup` command.

```
\ifimki@original
 \expandafter\def\expandafter\theindex\expandafter{\expandafter
    \imki@maybeaddtotoc\theindex}
\else
  \global\let\imki@idxprologue\relax
  \RequirePackage{multicol}
  \renewenvironment{theindex}
    {\imki@maybeaddtotoc
     \imki@indexlevel{\indexname}\imki@indexheaders
     \thispagestyle{\imki@firstpagestyle}%
     \ifnum\imki@columns>\@ne
       \columnsep \imki@columnsep
       \ifx\imki@idxprologue\relax
         \begin{multicols}{\imki@columns}
       \else
         \begin{multicols}{\imki@columns}[\imki@idxprologue]
       \fi
     \else
       \imki@idxprologue
     \fi
     \global\let\imki@idxprologue\relax
     \parindent\z@
     \parskip\z@ \@plus .3\p@\relax
     \columnseprule \ifKV@imki@columnseprule.4\p@\else\z@\fi
     \raggedright
     \let\item\@idxitem
     \imki@othercode}
    {\ifnum\imki@columns>\@ne\end{multicols}\fi
% \clearpage
}
\fi
```

The command `\indexsetup` may be used to customize some aspects of index formatting.

```
\def\imki@indexlevel{%
  \@ifundefined{chapter}{\section}{\chapter}*}
\define@key{imkiindex}{level}{\def\imki@indexlevel{#1}}
\def\imki@toclevel{%
  \@ifundefined{chapter}{section}{chapter}}
\define@key{imkiindex}{toclevel}{\def\imki@toclevel{#1}}
\define@boolkey{imkiindex}{noclearpage}[true]{\let\imki@clearpage\relax}
\def\imki@indexheaders{%
  \@mkboth{\MakeUppercase\indexname}{\MakeUppercase\indexname}}
```

```
\define@key{imkiindex}{headers}{\def\imki@indexheaders{\markboth#1}}
\def\imki@firstpagestyle{plain}
\define@key{imkiindex}{firstpagestyle}{\def\imki@firstpagestyle{#1}}
\let\imki@othercode\relax
\define@key{imkiindex}{othercode}{\def\imki@othercode{#1}}
\newcommand{\indexsetup}[1]{%
  \ifimki@original\else\setkeys{imkiindex}{#1}\fi}
\@onlypreamble\indexsetup
```

The command `\indexprologue` sets the internal version which is always `\let` to `\relax` during `\begin{theindex}`.

```
\newcommand{\indexprologue}[2][\bigskip]{%
  \long\gdef\imki@idxprologue{{#2\par}#1}}
```

Now we provide the relevant `\printindex` macros by transferring the real job to a secondary macro `\imki@putindex` after due checks and messages.

```
\providecommand*{\printindex}{}
\renewcommand*{\printindex}[1][\jobname]{%
  \@ifundefined{#1@idxfile}{\imki@error{#1}}{\imki@putindex{#1}}}

\def\imki@error#1{%
  \def\@tempa{#1}\def\@tempb{\jobname}%
  \ifx\@tempa\@tempb
    \let\imki@optarg\@empty
  \else
    \def\imki@optarg{[#1]}%
  \fi
  \PackageError{imakeidx}
    {Misplaced \protect\printindex\imki@optarg}
    {You are not making this index, as no appropriate
     \protect\makeindex\MessageBreak
     command has been issued in the preamble.}}
```

We define a command to do a `\cleardoublepage` if the option *openright* holds (in classes where *twoside* is meaningful). In case `\chapter` is defined but not `\if@openright`, we assume that the class wants "open right".

```
\def\imki@clearpage{%
  \@ifundefined{chapter}
    {\clearpage} % article and similar classes
    {\@ifundefined{if@openright}
       {\cleardoublepage}
       {\if@openright
          \cleardoublepage
        \else
          \clearpage
       \fi}
    }}
```

We need a helper macro to do a check in order to avoid a loop and the hook where to insert the table of contents related stuff.

```
\def\imki@check@indexname{\indexname}
```

```
\providecommand*\imki@maybeaddtotoc{}
```

Two helper macros for preparing the final messages to the user.

```
\def\imki@finalmessage#1{%
  \expandafter\edef\csname imki@message#1\endcsname
    {\imki@program\imki@options#1.idx}
  \AtEndDocument{\PackageWarning{imakeidx}{%
    Remember to run \imki@engine\space again after calling\MessageBreak
    `\@nameuse{imki@message#1}'\imki@shellwarn\@gobble}}}
\def\imki@splitindexmessage{%
  \AtEndDocument{\PackageWarningNoLine{imakeidx}{%
    Remember to run \imki@engine\space again after calling\MessageBreak
    `splitindex' and processing the indices\imki@shellwarn}}}
```

Here is a helper macro for deciding whether to call the external utility or to issue a final message. In `\imki@makeindexname` we put the name of the only program allowed by default (*makeindex*). If the list is updated, we can supplement the list here, maybe defining a list macro; for now this is sufficient. The temporary switch `\if@tempswa` is set to true if automatic processing is possible, so that the main macro can take the appropriate action.

```
\def\imki@makeindexname{makeindex}
\def\imki@decide{%
  \@tempswafalse
  \ifimki@splitindex % splitindex is not "safe"
    \ifnum\imki@shellescape=\@ne\@tempswatrue\fi
  \else
    \ifx\imki@program\imki@makeindexname % nor is texindy
      \ifnum\imki@shellescape=\tw@\@tempswatrue\fi
    \fi
    \ifnum\imki@shellescape=\@ne\@tempswatrue\fi
  \fi
  \ifKV@imki@noautomatic
    \@tempswafalse
  \fi}
```

We now define the main macro that puts the specified index file into the document and possibly orders to add the index title to the table of contents. It is duplicated as usual. The argument `#1` is the specific symbolic name of the index. In particular if the *intoc* option has been specified, the hook `\imki@maybeaddtotoc` is defined in such a way that the relevant information is added to the `toc` file. The `\phantomsection` command is necessary when using hyperref; here it is hidden as argument to `\@nameuse`, so it is equivalent to `\relax` and does nothing if hyperref has not been loaded.

```
\def\imki@putindexsplit#1{%
  \ifimki@nonewpage\else
    \imki@clearpage
  \fi
  \immediate\closeout\csname #1@idxfile\endcsname
  \let\imki@indexname\indexname % keep \indexname
  \@nameuse{imki@set@#1}\imki@decide
```

```
  \if@tempswa % we can call the external program
    \imki@exec{\imki@program\imki@options#1.idx}%
  \else
    \imki@finalmessage{#1}%
  \fi
  \ifKV@imki@intoc
    \def\imki@maybeaddtotoc{\@nameuse{phantomsection}%
      \addcontentsline{toc}{\imki@toclevel}{\imki@title}}%
  \else
    \def\imki@maybeaddtotoc{}%
  \fi
  \ifx\imki@title\imki@check@indexname\else
    \def\indexname{\imki@title}%
  \fi
  \@input@{#1.ind}
  \let\indexname\imki@indexname % restore \indexname
}

\newif\ifimki@splitdone
\def\imki@putindexunique#1{%
  \ifimki@nonewpage\else
    \imki@clearpage
  \fi
  \let\imki@indexname\indexname % keep \indexname
  \@nameuse{imki@set@#1}\imki@decide
  \if@tempswa % we can call the external program
    \ifimki@splitdone\else
      \immediate\closeout\@indexfile
      \imki@exec{splitindex \imki@splitindexoptions\space\jobname.idx}%
      \global\imki@splitdonetrue
    \fi
  \else
    \ifimki@splitdone\else
      \imki@splitindexmessage\global\imki@splitdonetrue
    \fi
  \fi
  \if@tempswa % we can call the external program
    \imki@exec{\imki@program\imki@options\jobname-#1.idx}%
  \fi
  \ifKV@imki@intoc
    \def\imki@maybeaddtotoc{\@nameuse{phantomsection}%
      \addcontentsline{toc}{\imki@toclevel}{\imki@title}}%
  \else
    \def\imki@maybeaddtotoc{}%
  \fi
  \ifx\imki@title\imki@check@indexname\else
    \def\indexname{\imki@title}%
  \fi
  \@input@{\jobname-#1.ind}
  \let\indexname\imki@indexname % restore \indexname
```

```
}
```

At this point, we choose the meaning of the relevant commands, reclaiming the space occupied by the discarded ones

```
\ifimki@splitindex
  \let\imki@startidx\imki@startidxunique
  \let\@wrindex\imki@wrindexunique
  \let\imki@putindex\imki@putindexunique
  \let\imki@startidxsplit\@undefined
  \let\imki@wrindexsplit\@undefined
  \let\imki@putindexsplit\@undefined
\else
  \let\imki@startidx\imki@startidxsplit
  \let\@wrindex\imki@wrindexsplit
  \let\imki@putindex\imki@putindexsplit
  \let\imki@startidxunique\@undefined
  \let\imki@wrindexunique\@undefined
  \let\imki@putindexunique\@undefined
\fi
```

To end the code, we deal with memoir:

```
\@ifclassloaded{memoir}{\let\@wrindexm@m\@wrindex}{}
```

The end.

Change History

v1.0
General: First public version 1
v1.0a
General: Small bug correction . . . 1
v1.1
General: Fixed compatibility with memoir 1
Modified interaction with LuaTEX 1
v1.1a
General: Fixed bug with possibly defined \directlua 1
Fixed bug with possibly defined \directlua; now we leave the check to ifluatex; using also ifxetex for symmetry. 11