Creating More Than One Index Using splitidx And SplitIndex*

Markus Kohm†‡

2013/04/09

Abstract

With makeidx, there's a standard package in LaTeX to create one index for each document. But sometimes more than one index is needed. There are different packages with different solutions and different problems to generate multiple indices. splitidx implements another solution to this problem. In addition, splitidx also lets you customize the typesetting and appearance of these indices, as well as the formatting of individual index entries.

Contents

*This file is version v1.2a of file `splitidx.dtx`. Nevertheless it should be stable.

†Markus Kohm <komascript@gmx.info>

‡Many thanks to Michael Palmer who improved the English user manual of the SplitIndex.

1 Introduction

Standard LaTeX provides for only a single document index. To produce such an index, one usually loads makeidx and marks up index entries in the document using the `\index` command. When the document is processed with LaTeX, the `\index` commands from the document are written as `\indexentry` commands to the raw index file "`\jobname.idx`". The raw index file is then processed with MakeIndex or another auxiliary program like xindy, which will produce a sorted index file named "`\jobname.ind`". This file is then included at the end of the document using the `\printindex` command.[1]

The splitidx package extends this process to permit the creation of multiple indices. Separate indices are declared and given unique shortcut identifiers with the `\newindex` command. In the document, individual index entries are marked up and assigned to specific indices with the `\sindex` command. For each of the declared indices, a separate `.idx` file is generated, each of which is post-processed into a separate `.ind` file. These `.ind` files are then included in the document using a modified version of the `\printindex` command.

The process outlined thus far resembles that of other multi-index packages such as multind. The most straightforward way to implement this scheme, which is the one used by package multind and others, is to directly write a separate `.idx` file for each declared index when processing the document with LaTeX, and then to separately post-process each `.idx` file with MakeIndex. However, this approach can run into technical limitations. TeX can have no more than 16 files open for writing at any one time. Several of these file handles are required by LaTeX itself for other purposes, such as cross references, the table of contents, and possibly others, depending on the structure of your document. Therefore, if you need a large number of separate indices, the limited number of available file handles may become a problem. The splitidx package provides a solution to this problem.

If the number of indices can be accommodated within the number of available file handles, you can use splitidx with the package option `split`. Then, splitidx will directly write multiple raw index files, that is, it will behave according to the scheme just described. On the other hand, if the number of indices exceeds the number of available file handles, you can request splitidx to write all index entries to a single intermediate index file, which must then be post-processed in order to

[1] For further details, read [1] and e.g. [2].

obtain the separate raw index files. The post-processing of the intermediate file is done with the SplitIndex program, which exists in several different implementations (see below). This behavior of splitidx is activated by omitting option `split`, that is, it is the package's default behavior.

In addition to the construction of separate indices, splitidx also offers help with customizing the typesetting and appearance of these indices, as well as the formatting of individual index entries.

2 The SplitIndex program

2.1 Purpose

While the number of files TEX can open for writing is limited, using multiple indices is normally limited too. As already mentioned in section 1 this limitation may be neutralized using a single intermediate index file, that will be split into several raw index files by an external post-processor: SplitIndex.

2.2 Implementation

The program has been implemented in five different languages, as follows:

splitindex.pl This is written in perl. You need a perl interpreter to run it. If you are a Unix user, you have a perl interpreter and you can call splitindex.pl like you would call a binary program or a shell script from your shell. This is the reference implementation. I prefer to use this, because it was the first, the easiest and the shortest to be written.

splitindex.java This is written using Sun Java 1.4.1. I wrote it because Java is everywhere and may be installed everywhere and a lot of people are able to understand Java source files. Nevertheless There's no longer a precompiled version of this in the main distribution. But you may download it from the repository at http://sarovar.org/plugins/scmcvs/cvsweb.php/binaries/?cvsroot=splitindex

splitindex.c: This is a C source of splitindex. I wrote the C version because a lot of people like to have a binary and most software authors understand C, and some people want fast binaries instead of slow Java byte code — even, if the Java program is fast enough. Nevertheless, there are no longer binaries of generated from this source in the main distribution. But you may download some from the repository at http://sarovar.org/plugins/scmcvs/cvsweb.php/binaries/?cvsroot=splitindex

splitindex.tex: This is a TEX version of the program. Yes, you are right: it is a program written in TEX. It has not the whole functionality of the other programs (see subsection 3.14), but it is system-independent and you don't need to install any other program like perl or Sun Java 1.4. It is not impossible to fix all the disadvantages of this program — but it isn't easy and much more work than all the other programs.

splitindex.tlu: This is a new TEXLua version of the program. It is platform independent like the perl script. Note, that the syntax for regular expressions in Lua differs from the perl syntax, if you use it instead of the perl version. Distributors should prefere the perl version, if they also provide perl for the installation platform.

With the exception of the TEX version, all of these programs are also able to call the index processor on each of the resulting raw index files.

And where is the lisp, the smalltalk, the prolog, the ... version of splitindex? Hey, five languages are enough for me! If you need one more, write it!

3 Using the splitidx package

3.1 Setup

You can use splitidx as a drop-in-replacement for makeidx. If you do so, you just have to replace

```
\usepackage{makeidx}
```

by

```
\usepackage{splitidx}
```

`\makeindex` To activate index generation, you can use `\makeindex`, which is declared by the LATEX kernel. You can also load the package with the option `makeindex`:

```
\usepackage[makeindex]{splitidx}
```

which is almost the same like:

```
\usepackage{splitidx}\makeindex
```

Other package options are available. The effect of the `split` option was already described in section 1; further options are discussed below.

`\newindex` If you want to generate more than one index without shortcut, you should declare this using `\newindex` with syntax:

`\newindex[`⟨*index name*⟩`]{`⟨*shortcut*⟩`}`.

The mandatory argument ⟨*shortcut*⟩ is used to distinguish the different indices. See description of `\sindex` for more information about this. The optional argument ⟨*index name*⟩ is the name of the index. This is also the default heading of this index used by the macros `\printindex` and `\printsubindex` (see below). If you omit ⟨*index name*⟩, the shortcut will be used as index name.

While it is always good practice to declare all index explicitly in the preamble of the document, this *must* be done if you also use the package option `split`. In this case, the `\newindex` command opens a raw index file to write to for each declared index. As the only exception, the raw index file for the index entries with the default shortcut (`idx`) will be created automatically. As noted above,

the number of index files that you can create in this way is limited, which is due to the limited number of output streams provided by TEX. If you exceed this number, not only the `\newindex` macro itself may result in an error, but also `\tableofcontents`, `\listoffigures`, `\listoftables` and any other command that implicitly allocates an output stream.

A unique shortcut declared with `\newindex` to refer to a specific index becomes part of the filenames of the corresponding `.idx` and `.ind` files. Therefore, when you choose a shortcut, make sure that you only use characters or symbols in the ⟨*shortcut*⟩ that are allowed in filenames. On file systems that treat file names as case-insensitive, you should not mix uppercase and lowercase letters. For maximum portability and minimum hassle, it is best to always use only lowercase letters.

3.2 Marking up index entries

`\index` After loading the splitidx package, you may use the `\index` command to mark up index entries in your manuscript as usual. You can find the description of the argument and features of this command in reference [1]. The splitindex program (see subsection 3.10) will put all index entries that were produced with `\index` into the same raw index file, which is tagged with the unique shortcut "`idx`"; that is, the `\index` command does *not* allow you to assign index entries to separate indices. However, the `useindex` option allows you to change this behavior; this is discussed below.

`\sindex` The splitidx package also defines the command `\sindex` with the syntax:

`\sindex[`⟨*shortcut*⟩`]{`⟨*index-entry*⟩`}`

The `\sindex` command is splitidx' mechanism for placing individual index entries into specific indices. The target index is identified by passing its unique shortcut, as declared with `\newindex`, in the optional argument to `\sindex`. If not given, the shortcut defaults to "`idx`", which should therefore be used to identify some sort of general index.

If you like, you may also request that `\index` should be an alias for `\sindex`. To do so, you use the package option `useindex`, e.g.:

```
\usepackage[useindex]{splitidx}
```

This may be useful when using packages like jurabib that expect `\index` to be the index command.

3.3 Suppressing multiple index generation

Under some unfortunate circumstances, for example when working with a publisher that enforces a rigid document format, it may be necessary to merge the separate indices back into a single index. In this case, it is *not* necessary to strip out all the individually marked up index identifiers. Instead, you may load the splitidx package with the `allintoone` option:

```
\usepackage[allintoone]{splitidx}
```

or

```
\usepackage[allintoone,makeindex]{splitidx}
```

With this option, splitidx will do the stripping for you, that is, `\sindex[`⟨*shortcut*⟩`]{`⟨*indexentry*⟩`}` will be substituted with `\index{`⟨*indexentry*⟩`}` during LaTeX processing.

Note: Currently only one of the options ***allintoone*** *and* ***useindex*** *can be used at same time. If you try to use both,* ***useindex*** *will be disabled! This may result in many error messages!*

3.4 Customizing index entries

`\AtWriteToIndex` splitidx uses `\protected@write` to write the index entries to its output files. The `\AtWriteToIndex` macro lets you execute a piece of code each time an index is written to a specific index. Usage:

`\AtWriteToIndex{`⟨*shortcut*⟩`}{`⟨*code*⟩`}`

This may be useful if you want your index entries to reference not the page number but some other counter instead. For example, in order to make each index entry in the general index (identified by the `idx` shortcut) point to the corresponding section number, you would write

```
\AtWriteToIndex{idx}{\let\thepage\thesection}
```

Note that this will work only if the shortcut of the index is given explicitly in each marked-up index entry; for example,

```
\sindex[idx]{Roller blades}
```

instead of

```
\sindex{Roller blades}
```

Note, if you want to use command `\index` instead of `\sindex`, you should also use the package option `useindex`; without it, command `\index` will still write the page number to the index.

The `\AtWriteToIndex` command may be used only in the document preamble.

`\AtNextWriteToIndex` Sometimes it may be useful to execute some commands only for writing a single index entry. To do so, you may use

`\AtNextWriteToIndex{`⟨*shortcut*⟩`}{`⟨*commands*⟩`}`

instead of `\AtWriteToIndex`.

3.5 Automatic custom index commands

Some people do not like to write `\sindex[foo]{`⟨*entry*⟩`}`. They want to write `\foo{`⟨*entry*⟩`}`. For these people, the package option '`idxcommands`' has been implemented. This option defines a command with the name of the ⟨*shortcut*⟩ for each declared index. If you use this option, you'll get an error if a command with this name is already defined. Also note that if you are using this option, the characters of the shortcuts must be letters.

3.6 Preventing premature expansion of index entries

`\newprotectedindex`

When using the standard index package makeidx, the LaTeX kernel command `\index` may expand its argument. The kernel uses `\@sanitize` to avoid expansion in some cases. But this fails if the argument was already read by another macro. So if you define a macro that reads its argument, does something with it and then writes it to the index, this may expand the argument. For illustration, try the following:

```
\documentclass{article}
\usepackage{ngerman}
\usepackage{makeidx}\makeindex
\newcommand*{\Test}[1]{#1\index{#1}}
\begin{document}
\Test{"Anderung}
"Anderung\index{"Anderung}
\end{document}
```

This will result in two entries in the `.idx` file:

```
\indexentry{\active@dq  \dq@prtct{A}nderung}{1}
\indexentry{"Anderung}{1}
```

The first one is something expanded that is not wanted. Package splitindx behaves the same way by default. But if you use `\newprotectedindex` to define a new index, it uses a trick to avoid expansion. If all indices should behave like this, you may simply use the package option `protected`.

```
\documentclass{article}
\usepackage{ngerman}
\usepackage[protected,useindex,makeindex]{makeidx}
\newcommand*{\Test}[1]{#1\index{#1}}
\begin{document}
\Test{"Anderung}
"Anderung\index{"Anderung}
\end{document}
```

Will result in two entries at the `.idx` file:

```
\indexentry{"Anderung}{1}
\indexentry{"Anderung}{1}
```

If you want to know more about the trick, see the command `\@onelevel@sanitize` in the LaTeX kernel documentation, `source2e`.

3.7 Including the generated indices in your document

\printindex The \printindex command is used to print one index or all indices that are declared using \newindex. How it behaves depends on the syntax you are using.

Used like this:

\printindex[⟨*shortcut*⟩][⟨*index name*⟩]

the index file with the optional shortcut will be included and printed, with the optional ⟨*index name*⟩ being used as the title. If ⟨*index name*⟩ is omitted, the default index name declared with \newindex will be used instead. If this name was omitted as well, the shortcut itself will be used as the title.

If both optional arguments, ⟨*shortcut*⟩ and ⟨*index name*⟩, are omitted, and you are using simply

\printindex

this command behaves like \printindex from package makeidx. You should not use this if you are using multiple indices.

You may also print all indices that were declared using \newindex at once. Use the syntax:

\printindex*

to do so. The indices will be printed in the order you declared them using \newindex.

3.8 Typesetting the generated indices

\printindex uses the default index output of the class and the index processor you are using. Usually, this will be theindex environment, but it doesn't have to be this way. Note, however, that \printindex expects the name of the index to be contained in the \indexname macro; otherwise, it will fail to typeset the index name.[2]

\printsubindex The \printsubindex command is analogous to \printindex, but it performs some redefinitions before printing the index, as follows:

- demote the index heading level by 1, that is, format the index title using \section* instead of \chapter* with classes that define \chapter (such as book and report), and using \subsection* instead of \section* with classes that don't define \chapter (such as article);
- deactivate \onecolumn, \twocolumn and \clearpage, \cleardoublepage that are otherwise used to start a new page in each index,
- change the mark mechanism to use \markright instead of \markboth for setting up the running headers.

[2]This would be a failure of the class, not of the splitidx package. I don't know of any class with this failure.

Using this macro, you can print multiple indices in one chapter, if you are using a class with `\chapter`, or in one section, if you are using a class without `\chapter`.

`\setindexpreamble` If you are using a KOMA-Script class, you'll know this command. Package splitidx redefines this command as follows:

`\setindexpreamble[`⟨*shortcut*⟩`]{`⟨*preamble*⟩`}`

This allows you to define a separate preamble for each index. Note: Package splitidx doesn't print the preamble itself. Instead, before typesetting an index with a given shortcut using `\printindex` or `\printsubindex`, it assigns the user-defined preamble for this shortcut to the internal macro `\index@preamble`. At the user level, its value can be accessed with the `\useindexpreamble` macro (see below).

`\useindexpreamble` If you are defining your own index environment or if you extend the existing `theindex` environment using `\extendtheindex` or otherwise, you can use `\useindexpreamble` to retrieve the preamble previously defined for the current index using `\setindexpreamble`:

`\useindexpreamble[`⟨*additional commands*⟩`]`

This macro is not limited to the KOMA-Script classes; it can also be used e.g. with the standard classes. The commands passed in the optional argument ⟨*additional commands*⟩ are only used if the preamble for the current index is defined and not empty. Authors of wrapper classes may use this, e.g. to add additional vertical spaces after the index preamble if and only if an index preamble will be printed.

`\indexshortcut` The macro `\indexshortcut` is only defined within the body of `\printindex` and `\printsubindex`. It expands to the shortcut of the specific index that is being printed. It may be useful when defining your own index environment or extending the `theindex` environment using e.g. `\extendtheindex`.

`\extendtheindex` Most classes define the environment `theindex` to be used for printing the index. Using `\extendtheindex` with this syntax:

`\extendtheindex{`⟨*before begin*⟩`}{`⟨*after begin*⟩`}{`⟨*before end*⟩`}{`⟨*after end*⟩`}`

you may extend this environment. The commands passed in ⟨*before begin*⟩ are inserted in `\begin{theindex}` just after starting the group but before the existing code defined for this code block. The commands passed in ⟨*after begin*⟩ are inserted after `\begin{theindex}`. Analogously, the commands passed in ⟨*before end*⟩ are inserted before `\end{theindex}`, while those passed in ⟨*after end*⟩ are used within `\end{theindex}` just after ending the index but just before ending the group.

3.9 Examples

Let's see how you may get more than one index. The text of the example is silly, so don't think about the text, think about the usage of splitidx.

```
\documentclass{article} % We use article class ...
\usepackage{splitidx} % ... and the splitidx package
```

```
\makeindex % And we want index generation

% We define 4 indices:
\newindex[General Index]{idx} % Name and shortcut of the 1st index
\newindex[Index of Animals]{ani} % ... 2nd index
\newindex[Index of Fruits]{fru} % ... 3rd index
\newindex[Index of Vegetables]{veg} % ... 4th index

\begin{document}
Apples\sindex[fru]{apple} % an entry to fru index
and oranges\sindex[fru]{orange} % an entry to fru index
are fruits\sindex{fruits}. % an implicit entry to idx index
Tomatoes\sindex[veg]{tomato} % an entry to veg index
are
vegetables\sindex{vegetables}. % an implicit entry to idx index
Cats\sindex[ani]{cat} % an entry to ani index
are animals\sindex[idx]{animals}. % an explicite entry to idx index

\printindex* % print all indices
\end{document}
```

After processing the file above with LaTeX you'll get a raw index file with following contents:

```
\indexentry[fru]{apple}{1}
\indexentry[fru]{orange}{1}
\indexentry{fruits}{1}
\indexentry[veg]{tomato}{1}
\indexentry{vegetables}{1}
\indexentry[ani]{cat}{1}
\indexentry[idx]{animals}{1}
```

Section 3.10 explains how to convert this intermediate file into separate raw index files and index files. In the above example, all four index files are input with a single `\printindex*` command. Each file will produce a single section that start on a new page. The section headings "General Index", "Index of Animals", "Index of Fruits" and "Index of Vegetables" will be printed in onecolumn mode, followed by the index entries printed in twocolumn mode.

Maybe you would like to format all indices as subsections within one section. You can do this by replacing the `\printindex*` command in the example above with the following:

```
\twocolumn[% set the title onecolumn
 \section*{Indices} % the section with the indices%
 \markboth{Indices}{Indices} % setting up the running headline %
]% but the indices twocolumn
\printsubindex* % print all indices
```

Note that I've used `\printsubindex*` instead of `\printindex*` in this modified example.

We now turn to the running headers for the index pages. If you are using page style `plain`, which is default at article class, the running headers are empty, so you don't need to set them up. However, if you're using page style `headings` for your index pages and the `\section*` command to format the headings of the several indices, you should set up the running headers to match the current index titles. If you are using a KOMA-Script class, you can use `\addsec` or `\addsec*` instead of `\section*` to format the index titles, in which case you will not need to manually update the running headers.

Maybe you want the general index to be the section, while the other indices should be subsections of the general index. You might then try to replace the code above with the following:

```
%##### This will not do the thing you wanted!  #####
\printindex[idx] % print index idx as section
\printsubindex[ani] % print index ani as subsection
\printsubindex[fru] % print index fru as subsection
\printsubindex[veg] % print index veg as subsection
```

But this will result in a twocolumn section containing the general index (identified by `idx`) and three onecolumn subsections containing the other indices, and a page break after the general index. Why is this? LaTeX will switch to twocolumn mode as it enters the `theindex` environment (which is created by the `\printindex` command) and will revert to onecolumn mode when it exits `theindex`. If twocolumn mode was active before `\printindex`, a `\clearpage` command will be issued at the end of `theindex`. So what's the solution? Remembering the `\extendtheindex` command, you can write:

```
\begingroup % keep the following extension local to this group
 \extendtheindex% some changes of theindex environment
  {}% no change before beginning
  {}% no change after beginning
  {\let\onecolumn\relax % deactivate \onecolumn before ending
   \let\clearpage\relax % deactivate \clearpage before ending
  }% changes before ending
  {}% no change after ending
 \printindex[idx] % print index idx as section
\endgroup % end group with extended theindex environment
\printsubindex[ani] % print index ani as subsection
\printsubindex[fru] % print index fru as subsection
\printsubindex[veg] % print index veg as subsection
\onecolumn % finish the indices
```

With this extension, the whole index will be set in twocolumn mode, with no page break before the first subsection. However, you have to switch back to onecolumn mode manually at the end of the indices.

The example above may be modified as follows to obtain a onecolumn index:

```
\begingroup % hold following extension local to this group
 \makeatletter % allow @ at macro names
 \extendtheindex% some changes of theindex environment
  {\let\twocolumn\@firstoptofone % deactivate \twocolumn
   \let\onecolumn\@firstoptofone % deactivate \twocolumn
   \let\clearpage\relax % deactivate \clearpage
  }% changes before beginning
  {}% no change after beginning
  {}% no change before ending
  {}% no change after ending
 \makeatother % deactivate \makeatletter
 \printindex % print index
\endgroup % end group with extended theindex environment
```

This not only works with splitted indices but also with one single index.

I hope that these examples were useful to understand how to format indices using splitidx. The next section will show you how to generate separate indices from a single intermediate index file.

3.10 Splitting intermediate index files

Most commonly, it will be sufficient to call one of the splitindex programs with one parameter, the name of the intermediate index file. This will split the intermediate file into several raw index files, and then call MakeIndex on each of these. The program splitindex can be instructed to use another index processor such as xindy, or to pass additional options along to the index processor, e.g. "`-g`" to use German sorting with MakeIndex. While it may be a rare need, it is also possible to modify the parsing of the intermediate index file and the generation of the filenames and contents of the resulting raw index files.

The names of the options and the syntax of the Arguments is same at all of the programs except splitindex.tex (see subsection 3.14):

`--help`
`-h` Show information about usage, options and arguments and terminate without processing an index file.

`--makeindex` ⟨***program name***⟩
`-m` ⟨***program name***⟩ Call ⟨*program name*⟩ instead of makeindex to process each generated raw index file. You may set this variable to an empty value. How this may be done depends on the shell, which you are using. Using bash you may achieve an empty value using `""` or `''`. An empty value means that no index processor will be called on the generated raw index files.

`--identify` ⟨***regular expression***⟩
`-i` ⟨***regular expression***⟩ Uses ⟨*regular expression*⟩ to identify the index shortcut and the contents of the raw index file with this shortcut in the intermediate file. The default value is: "`^(\\indexentry)\[([^]]*)\](.*)$`" for all but splitindex.tlu. This means:

`^` Search from beginning of the line.

`(\\indexentry)`
: Search for "`\indexentry`" and set group 1 to this.

`\[` Search for "`[`" and ignore it.

`([^]]*)`
: Search for any character which is not "]" and set group 2 to this.

`\]` Search for "`]`" and ignore it.

`(.*)$`
: Search for all characters till end of line and set group 3 to these.

The ⟨*regular expression*⟩ is POSIX 1003.2 compatible. For splitindex.tlu the default is: "`^(\\indexentry)%[([^]]*)%](.*)$`".

`--resultis` ⟨***pattern***⟩
`-r` ⟨***pattern***⟩ Set the lines, which are written to the generated raw index files after identification (see option `--identify`) to ⟨*pattern*⟩. Each `$`⟨*digit*⟩ at ⟨*pattern*⟩ will be replaced by the corresponding group, e.g. `$1` will be replaced by the first group (see `--identify`). The default is: "`$1$3`" for all but splitindex.tlu resp. "`%1%3`" for splitindex.tlu, which means: contents of group 1 and group 3.

If the ⟨*regular expression*⟩ of option `--identify` doesn't match a line at the raw index file the line itself will be written.

`--suffixis` ⟨***pattern***⟩
`-s` ⟨***pattern***⟩ Set the suffix of the names of the generated raw index files after identification (see option `--identify`) to ⟨*pattern*⟩. Each `$`⟨*digit*⟩ at ⟨*pattern*⟩ will be replaced by the corresponding group, e.g. `$1` will be replaced by the first group (see `--identify`). The default is: "`-$2`" resp. "`-%2`", which means: character '`-`' followed by contents of group 2.

If the ⟨*regular expression*⟩ of option `--identify` doesn't match a line at the raw index file, all groups will be set to "`idx`".

`--verbose`
`-v` Increase verbosity by one. More verbose means: tell the user more about, what the program is doing.

`--version`
`-V` Show information about program version and terminate without processing a index file.

Some of the binaries compiled from the C source won't understand the long option names (`--makeindex`, `--identify` ...). In this case you'd have to use the alternative short option names (`-m`, `-i` ...).

The first non-option-argument in the command line is used as the name of the intermediate index file to be processed. All arguments that follow the argument "`--`" are interpreted as non-option arguments. All but the first non-option-arguments will be passed to the index processor.

You will find some examples in the following subsections.

3.11 Using splitindex.pl

This is the reference implementation. Let's use an example to demonstrate its use. If you have the following LaTeX file "allabout.tex":

```
\documentclass{article}
\usepackage[makeindex]{splitidx}
\begin{document}
  Apples\sindex[fru]{apple} and oranges\sindex[fru]{orange} are
  fruits\sindex{fruits}.
  Tomatos\sindex[veg]{tomato} are vegetables\sindex{vegetables}.
  Cats\sindex[ani]{cat} are animals\sindex[idx]{animals}.
\end{document}
```

this generates the intermediate index file "Fileallabout.idx":

```
\indexentry[fru]{apple}{1}
\indexentry[fru]{orange}{1}
\indexentry{fruits}{1}
\indexentry[veg]{tomato}{1}
\indexentry{vegetables}{1}
\indexentry[ani]{cat}{1}
\indexentry[idx]{animals}{1}
```

This file can't be processed by an index processor like MakeIndex. In order to split this intermediate file into several raw index files and run the default index processor, you do the following call (the $ is a symbol for the shell prompt):

```
$splitindex.pl allabout.idx
```

You may omit the extension ".idx":

```
$splitindex.pl allabout
```

Both commands will result in a file allabout-fru.idx:

```
\indexentry[fru]{apple}{1}
\indexentry[fru]{orange}{1}
```

a file allabout-idx.idx

```
\indexentry{fruits}{1}
\indexentry{vegetables}{1}
\indexentry{animals}
```

a file allabout-veg.idx:

```
\indexentry[veg]{tomato}{1}
```

and a file allabout-ani.idx:

```
\indexentry[ani]{cat}{1}
```

After generation of these files, it calls the default index processor using the command lines:

```
makeindex allabout-fru.idx
makeindex allabout-idx.odx
makeindex allabout-veg.idx
makeindex allabout-ani.idx
```

These calls create the index files `allabout-fru.ind`, `allabout-idx.ind`, `allabout-veg.ind` and `allabout-ani.ind`, which can be loaded into the document using e.g. `\printindex` from package splitidx.

If you don't want splitindex to call any index processor, use

```
$splitindex.pl -m "" allabout
```

instead of the shell command above.

You may obtain the same files as above using (it's one input line not two as shown here):

```
$splitindex.pl -i '^\\indexentry\[([^]]*)\](.*)$' -s '-$1'
-r '\\indexentry$2' allabout
```

If you want splitindex to call makeindex with the additional option "`-s foo.ist`" to make it use the MakeIndex style file `foo.ist`, you can do so as follows:

```
$splitindex.pl allabout -- -s foo.ist
```

As you can see "`--`" is used to prevent splitindex from interpreting "`-s foo.ist`" as option "`--suffixis foo.ist`". All splitindex options must be put before "`--`", but you can put the raw file argument "`allabout`" after that:

```
$splitindex.pl -- allabout -s foo.ist
```

If you want so use the index processor xindy instead of default index processor MakeIndex, you can use this call:

```
$splitindex.pl -m xindy allabout
```

If xindy is not in your standard `PATH`, you may set the whole path:

```
$splitindex.pl -m /home/me/bin/xindy allabout
```

With most perl implementations, the perl module `Getopt::Long` allows to put options after no-option-arguments. So you may also write:

```
$splitindex.pl allabout -m /home/me/bin/xindy
```

with the same result.

3.12 Using splitindex.jar

This implementation should also be portable. If you are not using Sun Java 1.4.1 or higher, you may try to recompile this using the shell command:

```
$javac splitindex.java
```

This should result in a new `splitindex.class`. But it will fail e.g. with Sun Java 1.3, because regular expressions are needed, which are not available in Sun Java 1.3.

The call of splitindex.class is almost the same as shown for subsection 3.11 for splitindex.pl, but you have to replace "`splitindex.pl`" by "`java splitindex`". So the last example from subsection 3.11 becomes:

```
$java splitindex allabout -m /home/me/bin/xindy
```

3.13 Using splitindex or splitindex.exe

The Linux program splitindex was compiled using glibc, so it works the same as splitindex.pl and you may use not only:

```
$splitindex -m /home/me/bin/xindy allabout
```

but also:

```
$splitindex allabout -m /home/me/bin/xindy
```

But the CygWin program splitindex.exe was compiled using a CygWin library. Because of this, all options must be put before the first non-option argument. So you have to use:

```
$splitindex.exe -m /home/me/bin/xindy allabout
```

With:

```
$splitindex.exe allabout -m /home/me/bin/xindy
```

the argument "`-m /home/me/bin/xindy`" will be passed to the default index processor MakeIndex!

You need the CygWin-DLL `cygwin1.dll` to run splitindex.exe. If you haven't already installed it, you may download the DLL from http://cygwin.com/snapshots. You need bzip2, which can be found at http://source.redhat.com/bzip2, to decompress it. Alternatively, you may use http://cygwin.com/setup.exe to download and install a minimal CygWin environment.

The Linux-i386-ELF binary splitindex was compiled and linked using:

```
$gcc -O3 -Wall -osplitindex splitindex.c
$strip splitindex
```

The gcc was:

```
gcc (GCC) 3.2
Copyright (C) 2002 Free Software Foundation, Inc.
```

The used glibc is version 2.1.

If you compile another binary e.g. for BSD, please contact me, so we may put the new binary into the distribution or can build another binary distribution.

3.14 Using splitindex.tex

The TEX or LATEX program splitindex.tex doesn't know any options or arguments. The number of files that it can generate is limited to to number of TEX's free write handles. If there are any other lines than "`\indexentry`"-lines in the raw index file, running splitindex.tex will result in an error.

You may use splitindex.tex interactively:

```
$tex splitindex
```

or

```
$latex splitindex
```

If you do so, you will be asked for the name of the raw index file. You have to omit the extension "`.idx`" answering that question.

You may also use the splitindex.tex not interactively, e.g. if you are working with a batch. To do so you have to define macro `\IDX` to the name of the raw index file without the extension "`.idx`". So the first example of subsection 3.11 would become:

```
$tex \def\IDX{allabout}\input splitindex
```

You may also use LATEX instead of TEX:

```
$latex \def\IDX{allabout}\input splitindex
```

The current version of splitindex.tex doesn't call any index processor. But maybe a future version will be able to do so.

3.15 Merging Indices

Now you should know how to use package splitidx and the SplitIndex programs to split the index. But what about combining two or more indices to one, e.g. you want vegetables and fruits in the same index? Try this:

```
\documentclass{article} % We use article class ...
\usepackage{splitidx} % ... and the splitidx package
\makeindex % And we want index generation

% We define 4 indices:
\newindex[General Index]{idx} % Name and shortcut of the 1st index
\newindex[Index of Animals]{ani} % ... 2nd index
\newindex[Index of Fruits And Vegetables]{fru} % ... 3rd index
```

```
\begin{document}
Apples\sindex[fru]{apple} % an entry to fru index
and oranges\sindex[fru]{orange} % an entry to fru index
are fruits\sindex{fruits}. % an implicit entry to idx index
Tomatoes\sindex[veg]{tomato} % an entry to veg index
are
vegetables\index{vegetables}. % an implicit entry to idx index
Cats\sindex[ani]{cat} % an entry to ani index
are animals\sindex[idx]{animals}. % an explicite entry to idx index

\printindex* % print all indices
\end{document}
```

And do the following call after splitting the index using SplitIndex:

```
$makeindex allabout-veg.idx allabout-fru.idx
```

Alternatively you can concatenate `allabout-fru.idx` to `allabout-veg.idx` before running the index processor on `allabout-veg.idx`.

4 Implementation of splitidx

1 ⟨*package⟩

4.1 Options

The first option is used to activate index generation.

```
2 \DeclareOption{makeindex}{\AtEndOfPackage{\makeindex}}
```

With option `useindex` the original command `\index` behaves like `\sindex`.

```
3 \DeclareOption{useindex}{%
4   \def\@se@nd@xc@d@{\let\index\sindex}%
5   \AtEndOfPackage{\@se@nd@xc@d@}%
6 }
7 \let\@se@nd@xc@d@\relax
```

There is also an option to make `\sindex` ignores the optional argument and behaves like `\index`.

```
8 \DeclareOption{allatone}{%
9   \PackageWarning{splitidx}{Option ‘allatone’ deprecated!\MessageBreak
10    You should replace it by ‘allintoone’}%
11  \ifx\@se@nd@xc@d@\relax\else
12    \PackageInfo{splitidx}{option ‘allatone’ overwrites option ‘useindex’}%
13    \let\@se@nd@xc@d@\relax
14  \fi
15  \AtEndOfPackage{%
16    \renewcommand*{\sindex}[1][]{\index}%
17    \g@addto@macro\makeindex{\renewcommand*{\sindex}[1][]{\index}}%
18  }%
19 }
```

```
\DeclareOption{allintoone}{%
  \ifx\@se@nd@xc@d@\relax\else
    \PackageInfo{splitidx}{option 'allintoone' overwrites option 'useindex'}%
    \let\@se@nd@xc@d@\relax
  \fi
  \AtEndOfPackage{%
    \renewcommand*{\sindex}[1][]{\index}%
    \g@addto@macro\makeindex{\renewcommand*{\sindex}[1][]{\index}}%
  }%
}
```

Do not expand index arguments.

```
\newif\if@verbindex\@verbindexfalse
\DeclareOption{protected}{\@verbindextrue}
```

With option `idxcommands` every `\newindex` also defines a new index command.

```
\newif\if@newidxcmd\@newidxcmdfalse
\DeclareOption{idxcommands}{\@newidxcmdtrue}
```

With option `split` each index uses its own index file.

```
\newif\if@splitidx\@splitidxfalse
\DeclareOption{split}{\@splitidxtrue}
```

Processing the options

```
\ProcessOptions\relax
```

4.2 Setting an Index Entry

`\see` `\seealso` `\seename` `\alsoname`

These are four standard macros, which are also defined at makeidx. Hey, these definitions are stolen from makeidx! No, no, I'm not a bad guy, read "`legal.txt`", which comes with makeidx.

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

`\sindex` `\@wrsindex` `\@@wrsindex`

This works similar to original `\index` but uses a splitted index. So it allows an optional argument.

```
\newcommand*{\sindex}[2][]{%
}
\g@addto@macro\makeindex{%
  \renewcommand*{\sindex}{%
    \@bsphack\begingroup
    \@sanitize
    \@wrsindex
  }%
  \typeout{Using splitted index at \jobname.idx}%
  \@se@nd@xc@d@
}
```

At the following `\@@wrsindex` is used as a hook. If it is defines, it is used to write out the index entry. This hook may be used from e.g. hyperref to add `hyperpage` to the font selection of the page number. This only works with `encap |`.

```
\newcommand*{\@wrsindex}[2][]{%
  \ifx\relax#1\relax
    \if@splitidx
      \@wrsindex[idx]{#2}%
    \else
      \def\@tempa{#2}%
      \if@verbindex\@onelevel@sanitize\@tempa\fi
      \@wrindex{\@tempa}%
    \fi
  \else
    \def\@tempa{#2}%
    \csname index@#1@hook\endcsname
    \expandafter\ifx\csname @@wrsindex\endcsname\relax
      \@@@wrsindex{#1}{{\@tempa}{\thepage}}%
    \else
      \def\@tempb{\@@wrsindex{#1}}%
      \expandafter\@tempb\@tempa||\\%
    \fi
    \endgroup
    \@esphack
  \fi
}
\newcommand*{\@@@wrsindex}[2]{%
  \begingroup
    \if@splitidx
      \expandafter\ifx\csname @indexfile@#1\endcsname\relax
        \PackageError{splitidx}{%
          Index entry for not existing index%
        }{%
          You've tried to set an index to index `#1', without
          defining\MessageBreak
          that index before using \string\newindex.\MessageBreak
          This is only allowed, if you are not using package option
          `split'.%
        }%
      \else
        \expandafter\protected@write\csname @indexfile@#1\endcsname{%
          \csname index@#1@writehook\endcsname
          \csname index@#1@writehook@once\endcsname
        }{%
          \string\indexentry#2%
        }%
      \fi
    \else
      \protected@write\@indexfile{%
        \csname index@#1@writehook\endcsname
```

```
      \csname index@#1@writehook@once\endcsname
    }{%
      \string\indexentry[#1]#2%
    }%
  \fi
\endgroup
}
```

If hyperref was loaded at \begin{document} and hyperref-option hyperindex isn't disabled, and the hook is not used, define it:

```
\AtBeginDocument{%
  \begingroup\expandafter\expandafter\expandafter\endgroup
  \expandafter\ifx\csname ifHy@hyperindex\endcsname\relax
  \else
    \csname ifHy@hyperindex\endcsname
      \expandafter\ifx\csname @@wrsindex\endcsname\relax
        \def\@@wrsindex#1#2|#3|#4\\{%
          \ifx\\#3\\%
            \@@@wrsindex{#1}{{#2|hyperpage}{\thepage}}%
          \else
            \def\Hy@temp@A{#3}%
            \ifx\Hy@temp@A\HyInd@ParenLeft
              \@@@wrsindex{#1}{{#2|#3hyperpage}{\thepage}}%
            \else
              \ifx\Hy@temp@A\HyInd@ParenRight
                \@@@wrsindex{#1}{{#2|#3hyperpage}{\thepage}}%
              \else
                \@@@wrsindex{#1}{{#2|#3}{\thepage}}%
              \fi
            \fi
          \fi
        }%
      \fi
    \csname fi\endcsname
  \fi
}
```

\AtWriteToIndex Add commands to the write hook.

```
\newcommand*{\AtWriteToIndex}[1]{%
  \expandafter\ifx\csname index@#1@writehook\endcsname\relax
    \expandafter\let\csname index@#1@writehook\endcsname\@empty
  \fi
  \expandafter\g@addto@macro\csname index@#1@writehook\endcsname
}
```

\AtNextWriteToIndex Like \AtWriteToIndex only once.

```
\newcommand*{\AtNextWriteToIndex}[1]{%
  \expandafter\ifx\csname index@#1@writehook@once\endcsname\relax
    \expandafter\gdef\csname index@#1@writehook@once\endcsname{%
      \expandafter\global\expandafter\let\expandafter
```

```
      \csname index@#1@writehook@once\endcsname\relax
    }%
  \fi
  \expandafter\g@addto@macro\csname index@#1@writehook@once\endcsname
}
```

4.3 Printing One Or More Indices

\printindex
\printindex*

This is used to print an index in the normal way. In most cases this uses `theindex` environment, but it need not.

```
\newcommand*{\printindex}{%
```

The command may be called in the star version, which prints all defined indices. This is same as `\printindices`.

```
  \@ifstar {%
    \begingroup
      \let\printindex@@endhook=\printindex@endhook
      \let\printindex@endhook=\relax
      \printindices%
      \csname printindex@@endhook\endcsname
    \endgroup
  }{%
```

It may also be called with optional arguments to print one of the indices:

```
    \@ifnextchar [\@printindex%] brace check comment
```

Or it is called without any parameter and so it is same as at makeidx package:

```
    {%
      \@input@{\jobname.ind}%
      \csname printindex@endhook\endcsname
    }%
  }%
}
```

\@printindex

This is used to print one of the indices. The optional (here obligatory) argument is the shortcut of the index.

```
\newcommand*{\@printindex}{}
\def\@printindex[#1]{%
```

There can be one more optional argument, which is the title of the index. If not, the default title `\index@`⟨*shortcut*⟩`@name` is used.

```
  \@ifnextchar [%
  {\@@printindex[{#1}]}%
  {\@@printindex[{#1}][\csname index@#1@name\endcsname]}%
}
```

\@@pintindex

We use the default environment to print one of the indices, but we redefine `\indexname` to the title of the wanted index, `\indexshortcut` to the shortcut of the wanted index and `\index@preamble` to the preamble of the wanted index. We do this in a group so it is local.

```
\newcommand*{\@@printindex}{}
\def\@@printindex[#1][#2]{%
  \begingroup
    \edef\indexshortcut{#1}%
    \def\indexname{#2}%
    \let\index@preamble\relax
    \expandafter\let\expandafter\index@preamble
    \csname index@\indexshortcut @preamble\endcsname
    \if@splitidx
      \def\@tempa{idx}\def\@tempb{#1}%
      \ifx\@tempa\@tempb\let\@indexsuffix\@gobble\fi
    \fi
    \@input@{\jobname\@indexsuffix{#1}.ind}%
  \endgroup
  \csname printindex@endhook\endcsname
}
```

`\@indexsuffix` This generated the suffix from the shortcut. You may redefine this function, if you need. I'm using a trick here, to define the macro with proper catcodes but not to define it global. You may also use `\@firstofone` instead of `\lowercase`.

```
\begingroup
\catcode`\-12
\lowercase{\endgroup
  \newcommand*{\@indexsuffix}[1]{-#1}%
}
```

`\printindices` This is used to print all defined indices in the order of their definition and with their default titles. If the list is empty, is behaves like `\printindex` without star and optional arguments.

```
\newcommand*{\printindices}{%
  \ifx\@indices\@empty
    \printindex
  \else
    \begingroup
      \@for\@tempa:=\@indices\do{%
        \expandafter\printindex\expandafter[\@tempa]%
      }%
    \endgroup
  \fi
}
```

`\newindex` The definition of a new index has an obligatory argument, the shortcut for this index, and an optional argument, the name of this index. If you omit the optional argument the shortcut is used for the default name if the index. The definition will be done global!

```
\newcommand*{\newindex}[2][\relax]{%
  \@ifundefined{index@#2@name}{%
    \if@verbindex
      \expandafter\gdef\csname index@#2@hook\endcsname{%
```

```
        \@onelevel@sanitize\@tempa
      }%
    \else
      \expandafter\gdef\csname index@#2@hook\endcsname{}%
    \fi
    \ifx\@indices\@empty
      \xdef\@indices{#2}%
    \else
      \xdef\@indices{\@indices,#2}%
    \fi
    \ifx \relax#1
      \expandafter\xdef\csname index@#2@name\endcsname{#2}%
    \else
      \expandafter\xdef\csname index@#2@name\endcsname{#1}%
    \fi
    \if@newidxcmd
      \expandafter\newcommand\expandafter*\csname #2\endcsname{}%
      \expandafter\gdef\csname #2\endcsname{%
        \sindex[#2]%
      }%
    \fi
    \if@splitidx
      \def\@tempa{#2}\def\@tempb{idx}%
      \ifx\@tempa\@tempb
        \global\let\@indexfile@idx=\@indexfile
      \else
        \expandafter\newwrite\csname @indexfile@#2\endcsname
        \expandafter\immediate\expandafter\openout
        \csname @indexfile@#2\endcsname=\jobname-#2.idx
      \fi
    \fi
  }{%
```

If the index is already defined, an error occurs:

```
    \PackageError{splitidx}{%
      index ‘#2’ already defined%
    }{%
      You have already defined an index with shortcut ‘#2’.\MessageBreak
      You can’t define a new index with the same shortcut. If you’ll continue
      \MessageBreak
      The new definition will be ignored.%
    }%
  }%
}
\if@splitidx
  \@onlypreamble\newindex
\fi
```

`\newprotectedindex` Same like `\newindex` but always define an index with protected arguments.

```
\newcommand*{\newprotectedindex}[2][\relax]{%
```

```
  \begingroup\@verbindextrue\newindex[{#1}]{#2}\endgroup
}
```

`\@indices` This macro stores a list of the index shortcuts. This is needed by e.g. `\printindices` and build by `\newindex`.

```
\newcommand*{\@indices}{}
\gdef\@indices{}
```

`\extendtheindex` Extend `theindex` by some macros called before starting the index, after starting the index, before stopping the index and after stopping the index. This may be used to change index behaviour. One additional change is done, which may be useful: before the index `\index@preamble` is set to `\index@`⟨*shortcut*⟩`@preamble`.

```
\newcommand{\extendtheindex}[4]{%
  \begingroup\expandafter\expandafter\expandafter\endgroup
  \expandafter\ifx\csname splitindex@theindex\endcsname\relax
    \let\splitindex@theindex=\theindex
    \let\endsplitindex@theindex=\endtheindex
  \fi
  \renewcommand*{\theindex}{%
    #1\splitindex@theindex #2%
  }%
  \renewcommand*{\endtheindex}{%
    #3\endsplitindex@theindex #4%
  }%
}
```

`\setindexpreamble` Set one of the splitted index preambles or the original one.

```
\newcommand{\splitindex@setip}{}
\let\splitindex@setip\setindexpreamble
\let\setindexpreamble\relax
\newcommand{\setindexpreamble}[2][]{%
  \ifx \relax#1\relax
    \begingroup\expandafter\expandafter\expandafter\endgroup
    \expandafter\ifx\csname splitindex@setip\endcsname\relax
      \@namedef{index@preamble}{#2}%
    \else
      \splitindex@setip{#2}%
    \fi
  \else
    \@namedef{index@#1@preamble}{#2}%
  \fi
}
```

`\useindexpreamble` Use the index preamble and optional add additional information after it, if it exists and if it is not empty:

```
\newcommand{\useindexpreamble}[1][]{%
  \begingroup\expandafter\expandafter\expandafter\endgroup
  \expandafter\ifx\csname index@preamble\endcsname\relax\else
    \ifx\index@preamble\@empty\else
```

```
      \index@preamble #1%
    \fi
  \fi
}
```

\printsubindex
\printsubindex*

Works like \printindex but changes some macros before to level down the headings at the index generation.

```
\newcommand*{\printsubindex}{%
  \begingroup
    \begingroup\expandafter\expandafter\expandafter\endgroup
    \expandafter\ifx\csname chapter\endcsname\relax
      \let\section\subsection
      \begingroup\expandafter\expandafter\expandafter\endgroup
      \expandafter\ifx\csname addsec\endcsname\relax\else
        \def\addsec{\setcounter{secnumdepth}{0}\subsection}%
      \fi
    \else
      \let\chapter\section
      \def\@makeschapterhead{\section*}
      \let\@makechapterhead\section
      \begingroup\expandafter\expandafter\expandafter\endgroup
      \expandafter\ifx\csname addchap\endcsname\relax\else
        \let\addchap\addsec
      \fi
    \fi
```

Also, \onecolumn and \twocolumn and even \clearpage must be disabled. The macros \onecolumn and \twocolumn cannot be let \relax because the have an optional argument which must be used.

```
    \let\onecolumn\@firstoptofone
    \let\twocolumn\@firstoptofone
    \let\clearpage\relax
    \let\cleardoublepage\relax
```

And the mark mechanism must also use one down:

```
    \def\markboth{\expandafter\markright\@gobble}%
    \ifx\@mkboth\@gobble\else\let\@mkboth\markboth\fi
```

And the page style shouldn't change too:

```
    \let\thispagestyle\@gobble
```

Now, using \printindex enables all of it's features:

```
    \let\printindex@endhook=\endgroup
    \printindex
}
```

\@firstoptofone

Read the optional argument and do it.

```
\providecommand{\@firstoptofone}[1][]{#1}
```

⟨/package⟩

References

[1] Leslie Lamport: *MakeIndex: An Index Processor For LaTeX*, 17 February 1987

[2] Pehong Chen, Rick P. C. Rodgers: *MAKEINDEX(1L)*, Manual page, 10 December 1991

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.

Change History