Creating More Than One Index Using splitidx And SplitIndex*

Markus Kohm†

2006/03/07

Abstract

With makeidx there's a standard package at LaTeX to create one index to each document. But some times more than one index is needed. There are different packages with different solutions and different problems to achieve multiple indices. Here is one more.

Contents

1 Introduction

First of all you have to know how index generation normally would be done. Read [1] and e.g. [2] if you don't.

Generally you load a package e.g. makeidx, put `\index` commands into your document, which are written as `\indexentry` commands to the raw index file

*This file is version v0.9 of file `splitidx.dtx`. Nevertheless it should be stable.

†Markus Kohm <kohm@gmx.de>

"`\jobname.idx`". Then you call an *index processor* like MakeIndex or xindy, which generates a sorted index file "`\jobname.ind`". This will be included with a command like `\printindex` at the end of your document.

Most packages, which allows more than one index, open more than one raw index file. Each of these files costs a write file handle. TEX has only 16 of these. LATEX itself needs some of these for e.g. `.aux`, `.toc`, `.lot`, `.lof` and maybe other more or less temporary files depends on what you are doing. With package option `split` package splitidx works like those packages.

Without option `split` the splitidx package needs only one raw index file like makeidx would. This one raw index files contains all the index entries for all the indices you defined. But to do so splitidx and the index processor need help. Before calling the index processor the one raw index file has to be splitted into multiple raw index files. Each of these contains the index entries of one index. This splitting can be done with the splitindex program.

There is not only one SplitIndex program, there are SplitIndex programs in different programming languages:

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 prefers to use this — because it was the first, the easiest and the shortest to be written.

splitindex.jar This is written using Sun Java 1.4.1. You can find the source at `splitindex.java`. I wrote it, because Java is everywhere and may be installed everywhere and a lot of people are able to understand Java source files. But if you don't have Sun Java 1.4 the start of this program will result in errors — e.g.

```
Exception in thread "main" java.util.zip.ZipException:  No
such file or directory
```

if you try to start it with Sun Java 1.3.

splitindex-Linux-i386 This is a Linux-i386-ELF binary from the C source file `splitindex.c`. It will be renamed to splitindex during installation, if you are using Linux-i386. At section 3.3 you find a introduction in how to compile `splitindex.c` yourself e.g. if you are using Linux-PPC. I wrote the C source, 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 bit code — even, if the Java program is fast enough.

splitindex-OpenBSD-i386 This is a OpenBSD-i386 binary from the C source file `splitindex.c`. It will be renamed to splitindex during installation, if you are using OpenBSD-i386. It is almost the same like splitindex-Linux-i386 — but it doesn't understand long option names.

splitindex.exe This is a CygWin binary from the C source file `splitindex.c`. A CygWin binary is a Windows (Win32) binary using the CygWin DLL. At section 3.3 you find a introduction in how to use a CygWin binary without installing a whole CygWin environment — nevertheless it is nice to have a almost complete CygWin.

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 section 3.4), but it is system independent and you don't need to install a 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 program.

Without the TEX version of splitindex all of these programs are also able to call the index processor after creating the raw index files.

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

2 Using the splitidx package

2.1 Generating a raw index

You may 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 may use `\makeindex`, which is declared by the LATEX kernel. You may also load the package with option `makeindex`:

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

which is almost the same like:

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

`\index` After loading the package you may use command `\index`. You can find the description of the argument and features of this command at [1]. The splitindex programs (see section 3) put all index entries, which are produced with `\index` to the index with shortcut "`idx`".

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

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

to be used to put an index entry into the index with the optional given shortcut. If you omit optional ⟨*shortcut*⟩ the shortcut "`idx`" will be used. If you use `\sindex` you need to split the raw index file into several raw index files, before generating the index with an index processor like MakeIndex. See section 3 for more information about splitting the raw index file.

The shortcut is used not only to distinguish between the different indices. It is also used as part of the name of the raw index file generated by a splitindex program (see section 3) and the corresponding index file generated by an index processor like MakeIndex (see [2]). So you should not use characters or symbols at ⟨*shortcut*⟩ which are not allowed at filenames. At file systems, which are not case sensitive, you should not mix upper and lower case letters. Best would be, if you'll use only lower case letters.

Under some unfortunate circumstances you may be forced to put all your index entries back to one index. The easiest way to do this without changing all `\sindex` commands into `\index` commands is the package option `allatone`. If you load the package with:

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

or

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

`\sindex[`⟨*shortcut*⟩`]{`⟨*indexentry*⟩`}` will result in `\index{`⟨*indexentry*⟩`}`.

If you like, you may also declare that `\index` should be the same like command `\sindex`. In this case, you may use package option `useindex`, e.g.:

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

This may be usefull using packages like jurabib that expects `\index` to be the index command.

Note: Currently only one of the options `allatone` *and* `useindex` *can be used at same time. If you are using both* `useindex` *will be disabled! This may result in many error messages!*

2.2 Printing an index

`\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 from `\printindex` and `\printsubindex`. If you omit ⟨*index name*⟩ the shortcut will be used as index name.

Some people do not like to call an extra program like splitindex. For those the package option `split` has been implemented. With this option splitidx opens a new file for each index, which is declared using `\newindex`. If you are using this option, you have to declare all indices you want to use at the preamble of your document. This also uses the default `idx` file for index entries to index with shortcut `idx`. The number of index files, you can open is limited, if you are using this option. This is because of the limitation of output streams TEX provides. With this option not only `\newindex` may result in an error but also `\tableofcontents`, `\listoffigures`, `\listoftables` and each other command, which allocates an output stream.

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

`\newprotectedindex` Using standard index package makeidx the LATEX kernel command `\index` may expand the argument of `\index`. The kernel uses `\@santize` to avoid this in some

cases. But this fails, if the argument was already read e.g. by another macro. So if you define a macro, that reads an argument, does something with the argument and write it to the index this may expand the argument. Try 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 at 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 same by default. But if you are using `\newprotectedindex` to define a new index, it uses a trick so avoid expansion. If all indices should behave like this, you may simply use 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 command `\@onelevel@sanitize` at the LaTeX kernel documentation, `source2e`.

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

With syntax

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

the index file with the optional shortcut will be loaded and titled with optional given index name. If ⟨*index name*⟩ is omitted the default index name declared with `\newindex` will be used.

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

```
\printindex
```

it behaves like `\printindex` from package makeidx. You should not use this, if you are using `\sindex` with optional argument.

You may also print all indices, which were declared using `\newindex`. Use syntax:

```
\printindex*
```

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

`\printindex` uses the default index output of the class and the index processor you are using. Most this will be `theindex` environment, but it needn't. But `\printindex` will fail to set the name of the index if `\indexname` isn't use to print the name of the index. This would be a failure of the class not of the splitidx package. I don't know any class with this failure.

`\printsubindex` The `\printsubindex` command behaves like `\printindex` with same syntax but it does some redefinitions before printing the index, to:

- use `\section` instead of `\chapter` level at classes, which have `\chapter` and `\subsection` instead of `\section` level at classes, which haven't `\chapter`,
- deactivate `\onecolumn`, `\twocolumn` and `\clearpage`, `\cleardoublepage` to start a new page at each index,
- change the mark mechanism not to use `\markboth` but `\markright` for setting up the running headline.

Using this you can print multiple indices at one chapter, if you are using a class with `\chapter`, or at 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 to syntax:

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

So you can define a preamble for each index. Note: Package splitidx doesn't print the preamble itself. But it lets `\index@preamble` to be the preamble of the index with the actual shortcut, before printing an index using `\printindex` or `\printsubindex`.

`\useindexpreamble` If you are defining your own index environment or if you extend `theindex` environment using e.g. `\extendtheindex`, you may use `\useindexpreamble` with syntax:

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

to print the preamble of the actual index, which was set using `\setindexpreamble`. This is not related to the KOMA-Script classes, it can also be used e.g. with the standard classes. The commands from optional argument ⟨*additional commands*⟩ are only used, if the preamble is defined and not empty.

`\indexshortcut` The macro `\indexshortcut` is only defined at `\printindex` and `\printsubindex`. It expands to the shortcut of the actual index. So you may use it at 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 syntax

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

you may extend this command. The commands from ⟨*before begin*⟩ are used at `\begin{theindex}` just after starting the group but before starting the index. The commands from ⟨*after begin*⟩ are used after `\begin{theindex}`. The commands from ⟨*before end*⟩ are used before `\end{theindex}`. The commands from ⟨*after end*⟩ are used at `\end{theindex}` just after ending the index but just before ending the group.

2.3 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\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}
```

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 shows, how you can process this raw index file to get several raw index files and several index files. You will get four index files. Each of it will be input with the single `\printindex*` command at the example above. Each will produce a single section starting on an new page with one column section headings "General Index", "Index of Animals", "Index of Fruits" and "Index of Vegetables". Each index is printed in twocolumn mode.

Maybe you would like to have all indices being subsections at one section. You can do this, if you replace the `\printindex*` command at the example above by 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*` at this modification. You don't need to setup the running headline, if you are using page style `plain`, which is default at `article` class. But if you're using page style `headings` you should do this, if you are using `\section*`. If you are using a KOMA-Script class, you can use `\addsec` or `\addsec*` instead of `\section*` to not need manual updating of the running headline.

Maybe you want the general index to be the section, while the other indices should be subsections of the general index. Maybe you'll try to replace the above by 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 with general index `idx` and three onecolumn subsections with the other indices and a page break after the general index. Why? At the end of `theindex` environment of `\printindex` the onecolumn mode, which was valid before `\printindex` will be restored. If twocolumn mode was valid before `\printindex` a `\clearpage` command will be included at the end of `theindex`. So what's the solution? Remembering the `\extendtheindex` command you can write:

```
\begingroup % hold 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 twocolumn with no page break before the first subsection. But you have to switch back to onecolumn mode manually at the end of the indices.

The example above may be modified, if you want 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 index. You may use this also with one single index.

I hope, that these examples were useful to understand, how to use splitidx. Next section will show you, how to generate the indices from a single raw index.

3 Splitting the index

At most you'll call one of the splitindex programs with one parameter, the name of the raw index file, to split the raw index file into several raw index files and call the index processor MakeIndex. Some of you will also set options to use another index processor e.g. xindy or to set some options of the index processor e.g. "`-g`" to use German sorting with MakeIndex. Only few of you will also change the parsing of the raw index file and the generation if the filenames and contents of the several new 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 section 3.4):

`--help`
`-h` Show information about usage, options and arguments and terminate without processing a 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 e.g. using `""` or `''`. An empty value means: Don't call an index processor.

`--identify` ⟨***regular expression***⟩
`-i` ⟨***regular expression***⟩ Uses ⟨*regular expression*⟩ to identify the index short cut and the contents of the raw index file with this shortcut. The default value is: "`^(\\indexentry)\[([^]]*)\](.*)$`". 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.

`--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`", 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`", which means: character '`-`' followed by contents of group 2.

If the ⟨*regular expression*⟩ of option `--ifentify` 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.

The OpenBSD binary splitindex-OpenBSD-i386 doesn't understand the long option names (`--makeindex`, `--identify` ...). But you can use the alternative short option names (`-m`, `-i` ...).

The first no-option-argument at the command line is the name if the raw index file, which has to be processed. All arguments, which follow the argument "`--`" are interpreted as no-optional-arguments. All but the first no-option-arguments will be passed to the index processor.

You will find some examples at the following subsections.

3.1 Using splitindex.pl

This is the reference implementation. Let's use an example to demonstrate, how it works. If you have 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 a 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 a index processor like MakeIndex. If you want so split these raw index file into several 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 raw index files `allabout-fru.ind`, `allabout-idx.ind`, `allabout-veg.ind` and `allabout-ani.ind`, which can be loaded to 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 achieve the same files like above using (it's one input line not two like shown here):

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

If you want splitindex to call makeindex with additional options "`-s foo.ist`" to use the MakeIndex style file `foo.ist`, you can do this call:

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

As you see "`--`" is used to tell splitindex to not interprete "`-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 index processor xindy instead of default index processor MakeIndex you can do this call:

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

If this is not at the standard `PATH` you may set the whole path:

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

With most perl implementations 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.

3.2 Using splitindex.jar

This should also be portable. If you are not using Sun Java 1.4.1 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 at Sun Java 1.3.

The call of splitindex.class is almost the same like shown at section 3.1 for splitindex.pl, but you have to replace "`splitindex.pl`" by "`java splitindex`". So the last example from section 3.1 becomes:

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

3.3 Using splitindex or splitindex.exe

The Linux program splitindex was compiled using glibc, so it works same like 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 no-option-argument. So you have to use:

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

At:

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

the arguments "`-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. You may use `http://cygwin.com/setup.exe` to download and install a minimal CygWin environment alternatively.

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.

3.4 Using splitindex.tex

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

You may use splitindex.tex interactive:

```
$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 interactive 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 extension "`.idx`". So the first example of section 3.1 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 in future a version will be able to do so.

4 Combining 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 don't 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`.

5 Implementation of splitidx

1 ⟨*package⟩

5.1 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 it optional argument and behaves like `\index`.

```
\DeclareOption{allatone}{%
  \ifx\@se@nd@xc@d@\relax\else
    \PackageInfo{splitidx}{option 'allatone' 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
```

5.2 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
      \@@wrsindex{#1}\@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{}{%
          \string\indexentry#2%
        }%
      \fi
    \else
      \protected@write\@indexfile{}{%
        \string\indexentry[#1]#2%
      }%
    \fi
  \endgroup
}
```

If hyperref was loaded at `\begindocument` 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
}
```

5.3 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