NAME

makeindex − a general purpose, formatter-independent index processor

SYNOPSIS

makeindex [−c**] [**−g**] [**−i**] [**−l**] [**−o *ind*] **[**−p *num*] **[**−q**] [**−r**] [**−s *sfile*] **[**−t *log*] **[***idx0 idx1 idx2* . . .]

DESCRIPTION

The program *makeindex* is a general purpose hierarchical index generator; it accepts one or more input files (often produced by a text formatter such as TEX (*tex* (1L)) or *troff* (1), sorts the entries, and produces an output file which can be formatted. The index can have up to three levels (0, 1, and 2) of subitem nesting. The way in which words are flagged for indexing within the main document is specific to the formatter used; *makeindex* does *not* automate the process of selecting these words. As the output index is hierarchical, *makeindex* can be considered complimentary to the *awk* (1)-based *make.index* (1L) system of Bentley and Kernighan, which is specific to *troff* (1), generates non-hierarchical indices, and employs a much simpler syntax for indicating index entries. For illustration of use with *troff* and *TEX* , see the section EXAMPLES below.

The formats of the input and output files are specified in a style file; by default, input is assumed to be a *.idx* file, as generated by LATEX.

Unless specified explicitly, the base name of the first input file (*idx0*) is used to determine the names of other files. For each input file name specified, a file of that name is sought. If this file is not found and the file name has no extension, the extension *.idx* is appended. If no file with this name is found, *makeindex* aborts.

For important notes on how to select index keywords, see the document by Lamport cited below. As an issue separate from selecting index keywords, a systematic mechanism for placing index terms in a document is suggested in *Index Preparation and Processing* , a paper cited below.

OPTIONS

−c Compress intermediate blanks (ignoring leading and trailing blanks and tabs). By default, blanks in the index key are retained.

−g Employ German word ordering in the index, in accord with rules set forth in DIN 5007. By default, *makeindex* employs a word ordering in which precedence is: symbols, numbers, uppercase letters, lowercase letters. The sequence in German word ordering is: symbols, lowercase letters, uppercase letters, numbers. Additionally, this option enables *makeindex* to recognize the German TEX-commands {"a, "o, "u and "s} as {ae, oe, ue and ss} during the sorting of the entries. The quote character must be redefined in a style file (for example, redefine quote as '+'). If the quote character is not redefined, *makeindex* will produce an error message and abort.

−i Take input from *stdin* . When this option is specified and **−o** is not, output is written to *stdout* .

−l Letter ordering; by default, word ordering is used (see the ORDERING section).

−o *ind* Employ *ind* as the output index file. By default, the file name is created by appending the extension *.ind* to the base name of the first input file (*idx0*).

−p *num* Set the starting page number of the output index file to be *num* (useful when the index file is to be formatted separately). The argument *num* may be numerical or one of the following:

any The starting page is the last source page number plus 1.

odd The starting page is the first odd page following the last source page number.

even The starting page is the first even page following the last source page number.

The last source page is obtained by searching backward in the log file for the first instance of a number included within paired square brackets (**[**. . .**]**). If a page number is missing or the log file is not found, no attempt will be made to set the starting page number. The source log file name is determined by appending the extension *.log* to the base name of the first input file (*idx0*).

−q Quiet mode; send no messages to *stderr*. By default, progress and error messages are sent to *stderr* as well as to the transcript file.

−r Disable implicit page range formation; page ranges must be created by using explicit range operators; see SPECIAL EFFECTS below. By default, three or more successive pages are automatically abbreviated as a range (e.g. 1—5).

−s sty Employ *sty* as the style file (no default). The environment variable INDEXSTYLE defines the path where the style file should be found.

−t log Employ *log* as the transcript file. By default, the file name is created by appending the extension *.ilg* to the base name of the first input file (*idx0*).

STYLE FILE

The style file informs *makeindex* about the format of the *.idx* input files and the intended format of the final output file; examples appear below. This file can reside anywhere in the path defined by the environment variable INDEXSTYLE. The style file contains a list of <*specifier*, *attribute*> pairs. There are two types of specifiers: input and output. Pairs do not have to appear in any particular order. A line begun by '%' is a comment. In the following list of specifiers and arguments, <string> is an arbitrary string delimited by double quotes ("..."), <char> is a single letter embraced by single quotes ('...'), and <number> is a nonnegative integer. The maximum length of a <string> is 2048. A literal backslash or quote must be escaped (by a backslash). Anything not specified in the style file will be assigned a default value, which is shown at the head of the rightmost column.

INPUT STYLE SPECIFIERS

actual <char> ´@´
Symbol indicating that the next entry is to appear in the output file.

arg_close <char> ´}´
Closing delimiter for the index entry argument.

arg_open <char> ´{´
Opening delimiter for the index entry argument.

encap <char> ´|´
Symbol indicating that the rest of the argument list is to be used as the encapsulating command for the page number.

escape <char> ´\\´
Symbol which escapes the following letter, unless its preceding letter is **escape**. Note: **quote** is used to escape the letter which immediately follows it, but if it is preceded by **escape**, it is treated as a ordinary character. These two symbols *must* be distinct.

keyword <string> "\\indexentry"
Command which tells *makeindex* that its argument is an index entry.

level <char> ´!´
Delimiter denoting a new level of subitem.

quote <char> ´"´
Note: **quote** is used to escape the letter which immediately follows it, but if it is preceded by **escape**, it is treated as a ordinary character. These two symbols *must* be distinct.

range_close <char> ´)´
Closing delimiter indicating the end of an explicit page range.

range_open <char> ´(´
Opening delimiter indicating the beginning of an explicit page range.

OUTPUT STYLE SPECIFIERS

preamble <string> "\\begin{theindex}\n"
Preamble of output file.

postamble <string> "\n\n\\end{theindex}\n"
Postamble of output file.

setpage_prefix <string> "\n \\setcounter{page}{"
Prefix of command which sets the starting page number.

setpage_suffix <string> "}\n"
Suffix of command which sets the starting page number.

group_skip <string> "\n\n \\indexspace\n"
Vertical space to be inserted before a new group begins.

headings_flag <string> 0
Flag indicating treatment of new group headers, which are inserted when before a new group (symbols, numbers, and the 26 letters): positive values cause an uppercase letter to be inserted between prefix and suffix, and negative values cause a lowercase letter to be inserted (default is 0, which produces no header).

heading_prefix <string> ""
Header prefix to be inserted before a new letter begins.

symhead_positive <string> "Symbols"
Heading for symbols to be inserted if **headings_flag** is positive.

symhead_negative <string> "symbols"
Heading for symbols to be inserted if **headings_flag** is negative.

numhead_positive <string> "Numbers"
Heading for numbers to be inserted if **headings_flag** is positive.

numhead_negative <string> "numbers"
Heading for numbers to be inserted if **headings_flag** is negative.

item_0 <string> "\n \\item "
Command to be inserted between two primary (level 0) items.

item_1 <string> "\n   \\subitem "
Command to be inserted between two secondary (level 1) items.

item_2 <string> "\n   \\subsubitem "
Command to be inserted between two level 2 items.

item_01 <string> "\n   \\subitem "
Command to be inserted between a level 0 item and a level 1 item.

item_x1 <string> "\n   \\subitem "
Command to be inserted between a level 0 item and a level 1 item, where the level 0 item does not have associated page numbers.

item_12 <string> "\n   \\subsubitem "
Command to be inserted between a level 1 item and a level 2 item.

item_x2 <string> "\n   \\subsubitem "
Command to be inserted between a level 1 item and a level 2 item, where the level 1 item does not have associated page numbers.

delim_0 <string> ", "
Delimiter to be inserted between a level 0 key and its first page number (default: comma followed by a blank).

delim_1 <string> ", "
Delimiter to be inserted between a level 1 key and its first page number (default: comma followed by a blank).

delim_2 <string> ", "
Delimiter to be inserted between a level 2 key and its first page number (default: comma followed by a blank).

delim_n <string> ", "
Delimiter to be inserted between two page numbers for the same key in any level (default: comma followed by a blank).

delim_r <string> "--"
Delimiter to be inserted between the starting and ending page numbers of a range.

delim_t <string> ""
Delimiter to be inserted at the end of a page list. This delimiter has no effect on entries which have no associated page list.

encap_prefix <string> "\\"
First part of prefix for the command which encapsulates the page number.

encap_infix <string> "{"
Second part of prefix for the command which encapsulates the page number.

encap_suffix <string> "}".
Suffix for the command which encapsulates the page number.

line_max <number> 72
Maximum length of a line in the output, beyond which a line wraps.

indent_space <string> "\t\t"
Space to be inserted in front of a wrapped line (default: two tabs).

indent_length <number> 16
Length of **indent_space** (default: 16, equivalent to 2 tabs).

EXAMPLES

TEX EXAMPLE

The following example shows a style file called *book.ist*, which defines an index for a book which can be formatted independently of the main source:

```
preamble
"\\documentstyle[12pt]{book}
\\begin{document}
\\begin{theindex}
{\\small\n"
postamble
"\n\n}
\\end{theindex}
\\end{document}\n"
```

Assuming that a particular book style requires the index (as well as any chapters) to start from an odd page number, and that the input file is named *foo.idx*, the following command line produces output in file *footmp.ind*:

```
makeindex  −s book.ist  −o footmp.ind  −p odd  foo
```

Here a non-default output file name is used to avoid clobbering the output for the book itself (presumably *foo.dvi*, which would have been the default name for the index output file!).

TROFF EXAMPLE

A sample control file for creating an index, which we will assume resides in the file *sample.ist*:

```
keyword "IX:"
preamble
".\\\" start of index output
\".\\\" enter two column mode
.2C
.SH
.ce
INDEX
.XS
INDEX
.XE
.R
.ps 9p
.vs 11p
.sp
.de I1
.ti 0.25i
..
.de I2
.ti 0.5i
.."
postamble "\n.\\\" end of index output"
setpage_prefix "\n.nr % "
setpage_suffix ""
group_skip "\n.sp 1.0"
headings_flag 1
heading_prefix "\n.IS\n"
heading_suffix "\n.IE"
item_0 "\n.br\n"
item_1 "\n.I1\n"
item_2 "\n.I2\n"
item_01 "\n.I1\n"
item_x1 "\n.I1\n"
item_12 "\n.I2\n"
item_x2 "\n.I2\n"
delim_0 ", "
delim_1 ", "
delim_2 ", "
delim_r "-"
delim_t "."
encap_prefix "\\fB"
encap_infix ""
encap_suffix "\\fP"
indent_space ""
indent_length 0
```

The local macro package may require modification, as in this example of an extension to the −**ms** macros (note that at some sites, this macro should *replace* a pre-existing macro of the same name):

```
.
.de IX
.ie '\\n(.z'' .tm IX: \\$1 \\$2 \\$3 \\$4 \\$5 \\$6 \\$7 \\$8 \\$9 {\\n(PN}
```

```
.el \\!.IX \\$1 \\$2 \\$3 \\$4 \\$5 \\$6 \\$7 \\$8 \\$9 {\\n(PN}
..
```

(note that the string `{\\n(PN}` is separated from the rest of the line by a tab. If your local macro package does not contain this extension, just include those lines at the beginning of your file. Here is a simple *troff*(1) input file, which we will assume is named *sample.txt*:

```
This is a sample file to test the \fImakeindex\fP(1L)
program, and see
.IX {indexing!programs!C language}
.IX {makeindex@\fImakeindex\fP(1L)}
.bp
.rs
.IX {Knuth}
.IX {typesetting!computer-aided}
how well it functions in the \fItroff\fP(1) environment.
```

Note that index entries are indicated by the **.IX** macro, which causes the following text to be written to *stdout* along with the current page number.

CREATING THE INDEX FILE IN THE BOURNE SHELL

To create an input file for *makeindex*, **in the Bourne shell** environment, do the equivalent at your site of the command:

```
psroff -ms -Tpsc -t sample.txt > /dev/null 2> sample.tmp
```

Some sites will require *ditroff* instead of *psroff*. To filter out any genuine error messages, invoke *grep*(1):

```
grep '^IX: ' sample.tmp > sample.idx
```

CREATING THE INDEX FILE USING *UCSF* ENHANCED TROFF/TRANSCRIPT

With *UCSF* Enhanced troff/TRANSCRIPT, the **–I** option of *psroff*(1L) can produce both formatter output and an index file:

```
psroff -ms -I sample.inp -Tpsc sample.txt
```

If it is wished to suppress the formatter output:

```
psroff -ms -I sample.inp -Tpsc -t sample.txt > /dev/null
```

COMPLETING THE INDEX

Any of the above procedures leaves the input for *makeindex* in *sample.inp*. The next step is to invoke *makeindex*:

```
makeindex -s sample.ist sample.idx
```

This leaves *troff*(1)-ready output in the file *sample.ind*.

ORDERING

By default, *makeindex* assumes *word ordering*; if the **–l** option is in effect, *letter ordering* is used. In word ordering, a blank precedes any letter in the alphabet, whereas in letter ordering, it does not count at all. This is illustrated by the following example:

word order	*letter order*
sea lion	seal
seal	sea lion

Numbers are always sorted in numeric order. For instance,

9 (nine), 123
10 (ten), see Derek, Bo

Letters are first sorted without regard to case; when words are identical, the uppercase version precedes its lowercase counterpart.

A special symbol is defined here to be any character not appearing in the union of digits and the English alphabetic characters. Patterns starting with special symbols precede numbers, which precede patterns starting with letters. As a special case, a string starting with a digit but mixed with non-digits is considered to be a pattern starting with a special character.

SPECIAL EFFECTS

Entries such as

```
\indexentry{alpha}{1}
\indexentry{alpha!beta}{3}
\indexentry{alpha!beta!gamma}{10}
```

in the input file will be converted to

```
\item alpha, 1
   \subitem beta, 3
      \subsubitem gamma, 10
```

in the output index file. Notice that the **level** symbol ('!') is used above to delimit hierarchical levels.

It is possible to make an item appear in a designated form by using the **actual** ('@') operator. For instance,

```
\indexentry{alpha@{\it alpha\/}}{1}
```

will become

```
\item {\it alpha\/},  1
```

after processing. The pattern preceding '@' is used as sort key, whereas the one following it is written to the output file. Note that two appearances of the same key, one with and one without the **actual** operator, are regarded as *distinct* entries.

The item, subitem, and subsubitem fields may have individual sort keys:

```
\indexentry{aa@{\it aa\/}!bb@{\it bb\/}!cc@{\it cc\/}}{1}
```

This will be converted to

```
\item {\it aa}, 1
   \subitem {\it bb}, 3
      \subsubitem {\it cc}, 10
```

It is possible to encapsulate a page number with a designated command using the **encap** ('|') operator:

```
\indexentry{alpha|bold}{1}
```

will be converted to

```
\item alpha, \bold{1}
```

where, with a suitable definition for TEX, `\bold{n}` will expand to `{\bf n}`. In this example, the three output attributes associated with page encapsulation **encap_prefix**, **encap_infix**, and **encap_suffix**, correspond to backslash, left brace, and right brace, respectively. This mechanism allows page numbers to be set in different fonts. For example, the page where the definition of a keyword appears can be in one font, the location of a primary example can be in another font, and other appearances in yet a third font.

The **encap** operator can also be used to create cross references in the index:

```
\indexentry{alpha|see{beta}}{1}
```

will become

```
\item alpha, \see{beta}{1}
```

in the output file, where

```
\see{beta}{1}
```

will expand to

```
{\it see\/} beta
```

Note that in a cross reference like this the page number disappears.

A pair of **encap** concatenated with **range_open** ('|(') and **range_close** ('|)') creates an explicit page range:

```
\indexentry{alpha|(}{1}
\indexentry{alpha|)}{5}
```

will become

```
\item alpha, 1—5
```

Intermediate pages indexed by the same key will be merged into the range implicitly. This is especially useful when an entire section about a particular subject is to be indexed, in which case only the range opening and closing operators need to be inserted at the beginning and end of the section. Explicit page range formation can also include an extra command to set the page range in a designated font:

```
\indexentry{alpha|(bold}{1}
\indexentry{alpha|)}{5}
```

will become

```
\item alpha, \bold{1--5}
```

Several potential problems are worth mentioning. First, entries like

```
\indexentry{alpha|(}{1}
\indexentry{alpha|bold}{3}
\indexentry{alpha|)}{5}
```

will be interpreted as

```
\item alpha, \bold{3}, 1--5
```

but with a warning message in the transcript about encountering an inconsistent page encapsulator. An explicit range beginning in a Roman page number and ending in Arabic is also considered an error. In this instance, (if possible) the range is broken into two subranges, one in Roman and the other in Arabic. For instance,

```
\indexentry{alpha|(}{i}
\indexentry{alpha}{iv}
\indexentry{alpha}{3}
\indexentry{alpha|)}{7}
```

will be turned into

```
\item alpha, i--iv, 3--7
```

with a warning message in the transcript file complaining about an illegal range formation.

Finally, every special symbol mentioned in this section may be escaped by the **quote** operator ('"'). Thus

```
\indexentry{alpha"@beta}{1}
```

will actually become

```
\item alpha@beta,  1
```

as a result of executing *makeindex*. The quoting power of **quote** is eliminated if it is immediately preceded by **escape** ('\'). For example,

```
\indexentry{f\"ur}{1}
```

becomes

```
\item f\"ur, 1
```

which represents an umlaut-accented 'u' to the TEX family of processors.

From version 2.11 of *makeindex*, the **quote** operator may quote *any* character in the range 1 ... 255. Character 0 is excluded because it is used internally in the *makeindex* source code as a string terminator. With this change, sort keys can be created for all eight-bit characters except 0. The sorting order is

punctuation characters (in ASCII order),
digits,
control characters (1 . . . 31),
space (32),
letters (ignoring case),
characters 127 . . . 255.

Here is an example showing the indexing of all printable ASCII characters other than letters and digits, assuming the default TEX format. For convenience, the page number references are the corresponding ASCII ordinal values.

```
\indexentry{" @"  (space)}{32}
\indexentry{"!@"! (exclamation point)}{33}
\indexentry{""@"" (quotation mark))}{34}
\indexentry{"#@"\# (sharp sign)}{35}
\indexentry{"$@"\$ (dollar sign)}{36}
\indexentry{"%@"\% (percent sign)}{37}
\indexentry{"&@"\& (ampersand)}{38}
\indexentry{"<@"$<$ (left angle bracket)}{60}
\indexentry{"=@"= (equals)}{61}
\indexentry{">@"$>$ (right angle bracket)}{62}
\indexentry{"?@"? (query)}{63}
\indexentry{"@@"@ (at sign)}{64}
\indexentry{"[@"[ (left square bracket)}{91}
\indexentry{"\@"\verb=\= (backslash)}{92}
\indexentry{"]@"] (right square bracket)}{93}
\indexentry{"^@"\verb=^= (caret)}{94}
\indexentry{"_@"\verb=_= (underscore)}{95}
\indexentry{"`@"\verb=~= (grave accent)}{96}
\indexentry{"{@"\"{ (left brace)}{123}
\indexentry{"|@"\verb="|= (vertical bar)}{124}
\indexentry{"}@"\"} (right brace)}{125}
\indexentry{"~@"\verb=~= (tilde)}{126}
```

Characters in the actual fields following the '@' character which have special significance to TEX must be represented as control sequences, or as math mode characters. Note particularly how the entries for the at sign, left and right braces, and the vertical bar, are coded. The index file output by *makeindex* for this example looks like this:

```
\begin{theindex}

  \item ! (exclamation point), 33
  \item " (quotation mark)), 34
  \item \# (sharp sign), 35
  \item \$ (dollar sign), 36
  \item \% (percent sign), 37
  \item \& (ampersand), 38
```

```
    \item $<$ (left angle bracket), 60
    \item = (equals), 61
    \item $>$ (right angle bracket), 62
    \item ? (query), 63
    \item @ (at sign), 64
    \item [ (left square bracket), 91
    \item \verb=\= (backslash), 92
    \item ] (right square bracket), 93
    \item \verb=ˆ= (caret), 94
    \item \verb=_= (underscore), 95
    \item \verb=˜= (grave accent), 96
    \item \{ (left brace), 123
    \item \verb=|= (vertical bar), 124
    \item \} (right brace), 125
    \item \verb=˜= (tilde), 126

    \indexspace

    \item   (space), 32

  \end{theindex}
```

FILES

/usr/local/bin/makeindex	executable file
/usr/local/lib/tex/macros/idxmac-amstex.tex	TeX macro file used by *makeindex*
/usr/local/lib/tex/macros/idxmac.tex	TeX macro file used by *makeindex*
/usr/local/lib/tex/macros/makeidx.doc	TeX macro file used by *makeindex*
/usr/local/lib/tex/macros/makeidx.sty	TeX macro file used by *makeindex*

SEE ALSO

ditroff(1L), latex(1L), make.index (1L), qsort(3), tex(1L), troff(1L)

UCSF Enhanced troff/TRANSCRIPT — An Overview, R. P. C. Rodgers and Conrad Huang, LSMB Technical Report 90-2, UCSF School of Pharmacy, San Francisco, 1990.

Index Preparation and Processing, Pehong Chen and Michael A. Harrison, *Software: Practice and Experience*, **19**(9), 897–915, September 1988.

Automating Index Preparation, Pehong Chen and Michael A. Harrison. Technical Report 87/347, Computer Science Division, University of California, Berkeley, 1987 (a LaTeX document supplied with *makeindex*).

MakeIndex: An Index Processor for LaTeX, Leslie Lamport, February 1987 (a LaTeX document supplied with *makeindex*).

Tools for Printing Indices, Jon L. Bentley and Brian W. Kernighan, *Electronic Publishing — Origination, Dissemination, and Design*, 1(1), 3–18, June 1988 (also available as: Computing Science Technical Report No. 128, AT&T Bell Laboratories, Murray Hill, NJ 07974, 1986).

AUTHOR

Pehong Chen, Chen & Harrison International Systems, Inc. Palo Alto, California, USA <chen@renoir.berkeley.edu>.
Manual page extensively revised and corrected, and *troff*(1) examples created by Rick P. C. Rodgers, UCSF School of Pharmacy <rodgers@cca.ucsf.edu>.

ACKNOWLEDGMENTS

Leslie Lamport contributed significantly to the design. Michael Harrison provided valuable comments and suggestions. Nelson Beebe improved on the portable version, and maintains the source distribution for the TEX Users Group. Andreas Brosig contributed to the German word ordering. The modification to the **−ms** macros was derived from a method proposed by Ravi Sethi of AT&T Bell Laboratories. The *LOG* and *CONTRIB* files in the *makeindex* source distribution record other contributions.