Tips on Indexing LaTeX Documents[*]

Nelson H. F. Beebe

Contents

1 Introduction

Indexes can significantly enhance the value of technical documents, and tools like `makeindex` [4, 5] and `texindex` (from the Free Software Foundation) help to make the job of index preparation easier. Because a good index will have a large number of entries, each of which contains a page number that could change if the document is modified, even by as little as a single character, you should not even consider creating an index file by hand!

The *LaTeX User's Guide and Reference Manual* [11, pp. 77–79, 189], [12, pp. 74–75, 212] gives only brief instructions on preparation of indexes; the purpose of these notes is to provide some hints that may be helpful.

You can find additional guidance about index preparation in the documentation for `makeindex` [4, 5, 14, 15], as well as in Bentley and Kernighan's paper [3]. The `troff` indexing tools from the latter are available from `netlib` via the electronic mail request `send index from TYPESETTING` to `netlib@ornl.gov`, or via anonymous `ftp` and the World-Wide Web at `ftp://netlib.bell-labs.com/netlib/typesetting/indexingtools.gz`, which is a UNIX `shar` bundle.

More recently, Joachim Schrod's work on extensions to `makeindex` has evolved under his direction into the thesis work of Roger Kehr [6] (in German) and subsequent technical reports [7, 8, 9] (in English) on xîndy (flexible îndexing system), a completely new program. Although xîndy appears to have considerable power, especially for the issue of user-definable sorting rules, I shall not consider it further here, because the following discussion is mostly independent of indexing software, and because I have little experience yet with xîndy.

[*] This document was originally written in December 1991 and used locally; it has been updated here to reflect new developments since then.

2 LaTeX indexing macros

A LaTeX document to be indexed has this basic structure:

```
\documentclass[]{...}
\usepackage{makeidx}
\usepackage{...  ...}
...
\makeindex
\begin{document}
...
gnats%
\index{gnat}
and gnus%
\index{gnu}
...
Gnats%
\index{gnat!feeding habits}
...
In the summer, gnats%
\index{gnat!feeding habits!summer}
...
\index{wildebeest|see{gnu}}
\printindex
\end{document}
```

For old LaTeX 2.09 installations, replace the first two lines by

```
\documentstyle[makeidx]{...}
```

If index entries are put on separate lines for readability, as they are here, then it is *essential* that the previous line be ended with a percent following the last non-blank character. TeX discards text from percent to end of line, plus all following whitespace on the next line. Were a space to get between the phrase and its index entry, it is possible that a page break might fall there, resulting in an off-by-one page number in the index.

If you put multiple index entries on consecutive lines, end all but the last with percents. Otherwise, you risk getting unwanted space into the output; see [11, p. 153], [12, p. 169].

The `makeidx.sty` option that is read when the LaTeX `\documentstyle` command is processed is a simple one: it defines a `\see` macro for index cross-referencing, and a `\printindex` macro to input the index file, which is named `\jobname.ind`, where `\jobname` is the base name of the current LaTeX file.

The `\makeindex` command must be given before the `\begin{document}`; without it, no index file will be written.

While you are developing an index, it is useful to be able to tell what has been indexed on each page. If you include the `showidx` package (or for LaTeX 2.09, the option `showidx` in the `\document style` command), all of the index entries on the current page will be contained in a marginal note. When you have finished writing, and verified that the index is complete, you can simply remove the use of `showidx`.

Once the `\makeindex` command has been issued, an `\index{gnat!feeding habits}` call produces in a file with extension `.idx` an entry of the form

```
\indexentry{gnat!feeding habits}{27}
```

containing the text of the index entry, and the page number. It is the job of an indexing program, such as `makeindex`, to collect the entries, sort them, eliminate duplicates, merge page number ranges, and then output another file, with extension `.ind`, of the form

```
\begin{theindex}
...
 \indexspace

...
 \item gnat, 2, 11, 17--21
   \subitem feeding habits, 27
      \subsubitem summer, 28
...
\end{theindex}
```

The syntax of the argument of the `\index` macro depends on the indexing program that is used. This document assumes `makeindex`, since that is currently one of the most powerful and flexible indexing programs available, and it runs on all machines that TeX runs on.

Complete details of how to run `makeindex` and prepare index entries can be found in its UNIX manual page documentation. The examples above illustrate two of the major features:

- Up to 3 index levels can be given; they are separated by exclamation points in the `\index` argument.
- Vertical bar separates a topic from a cross-reference entry. Thus,

  ```
  \index{fourmis|voir{insectes}}
  ```

  would expand to the `.ind` file entry

  ```
  \item fourmis, \voir{insectes}{19}
  ```

  and the cross-reference macro would be defined to ignore its second argument (the page number):

  ```
  \newcommand{\voir}[2]{\emph{voir} #1}
  ```

  The `makeidx.sty` file defines the `\see` macro like this.

The page number generated by index cross-references like `\index{wildebeest|see{gnu}}` is ignored, so you can put such cross-references anywhere. It is a good idea to collect them all in one place, such as immediately before the `\printindex` command, so that you can easily find them. If you leave them scattered throughout the input file, inconsistencies are more likely to develop.

3 Typesetting the index

Because index entries that are written to a file by LaTeX must be processed by the indexing program, LaTeX must be run again to actually typeset the index. If you are also using BibTeX for bibliographies, and `\label`, `\pageref`, and `\ref` for cross-referencing, then the command sequence required in general looks like the one used for this document:

```
latex idxtips.ltx
bibtex idxtips
makeindex idxtips
latex idxtips.ltx
makeindex idxtips
latex idxtips.ltx
makeindex idxtips
```

If the bibliography entries themselves cite other entries, then you need to insert another BibTeX step after the second LaTeX step. In general, to obtain a correct index, `makeindex` must be run after each LaTeX step, because any change whatever in the typeset output could change a page number in an index entry.

Repeated issuing of commands like those above is tedious and error prone, and is best automated by putting them in a command script, or on UNIX, in a `Makefile`. Here is a suitable set of `Makefile` entries for doing this generically:

```
BIBTEX          = bibtex
LATEX           = latex
MAKEINDEX       = makeindex

.SUFFIXES:
.SUFFIXES: .dvi .ltx .ind .idx .bbl \
           .bib
```

```
.bib.bbl:
        - $(BIBTEX) $*

.idx.ind:
        $(MAKEINDEX) $*

.ltx.dvi:
        $(LATEX) $<
        $(MAKE) $*.bib
        $(MAKE) $*.ind
        $(LATEX) $<
        $(MAKE) $*.ind
        $(LATEX) $<
        $(MAKE) $*.ind
```

The reason for recursively invoking `make` in the commands for the rule `.ltx.dvi`, instead of running `bibtex` or `makeindex` directly, is that this makes it possible to later add specific targets that do extra things, such as the editing of `.idx` files described below.

Of course, while you are fixing minor typographical errors and solving formatting problems, you can manually run single LaTeX steps instead of the long command list; just remember to run the command script or `make` to do the job properly before you produce a version for printing.

4 Controlling the sort order and fonts

In simpler documents, the index may require only unaccented entries in roman text. More complex ones may make substantial use of font changes, or require accents in index entries [14, 15, 7, 8, 9]. Both of these complicate the sorting job that is required to turn the `.idx` file into a `.ind` file.

In the example above,

```
\index{gnat!feeding habits!summer}
```

the words `gnat`, `feeding habits`, and `summer` do two jobs: they define the *text* of the entry, and they define the *sort key*. `makeindex` provides for the separation of these two functions with the `@` (actual) character: `sort key@actual text`. This syntax may be used for any, or all, of the up-to-three levels of indexing. With a somewhat unusual choice of fonts, the sample entry might be then rewritten as

```
\index{gnat@{\bf gnat}!feeding
habits@{\em feeding
habits}!summer@{\sl summer}}
```

The font syntax here corresponds to plain TeX and LaTeX 2.09: it works equally well in newer LaTeX releases.

Accents and letters other than the Latin ‘a’ through ‘z’ pose somewhat more of a problem, because they generally require special sort ordering. For example, the Danish alphabet contains the letters ‘a’ through ‘z’, followed by three additional ones, ‘æ’, ‘ø’, ‘å’, represented in TeX by the control sequences `\ae`, `\o`, and `\aa`. In order to get these letters sorted properly, they must be represented in the sort key by characters which collate immediately after ‘z’. In such cases, `makeindex` version 3, or `xîndy`, will support the required sorting order.

The default sorting order in `makeindex` is

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

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 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.

A command line option to `makeindex` controls the handling of blanks in index entries; they may or may not be significant for sorting purposes.

5 Finding topics to index

Preparation of a good index is not an easy task. The person preparing the index must be able to pretend being a reader of the document who wants to find things in it, think up many variations on the words used in the index, and insert them all with suitable cross-referencing.

While it is usually best to wait until a document is almost complete before preparing the index entries, it is a good idea to make the job easier by doing some planning before you start writing or typing. Design some macros that make it easy to get phrases into the text and into the index, possibly under different index entries. A simple example is this macro:

```
\newcommand{\X}[1]{#1\index{#1}}
```

If you write `\X{gnats}`, then `gnats` will appear in both the text and the index. This case happens so frequently that it is worthwhile for the macro name to be short and easy to type. The LaTeX editing support developed by the author [1] for the Emacs text editor makes it easy to insert such entries in an X Window System environment: just press the right mouse button at the start of the phrase to be indexed, sweep the mouse across the phrase to somewhere in the last word, and release the mouse button.

To find such phrases, it is best to sit down in an easy chair with a typeset draft of the manuscript, and as you read it from start to finish, use a colored marker pen to highlight every phrase that should be indexed. You can then later go through the manuscript file on the computer to insert the entries. If you have the mouse button support described above, this is quite easy to do.

When font changes are involved, I have found it helpful in writing to use a pair of macros, one to place a phrase in the text and the index, and another just to index it. To allow flexibility, one such pair should be introduced for each type of entry. For example, in writing about computer software, I use these pairs:

```
% generate and index file name
\newcommand{\FILE}[1]
{%
  \texttt{#1}%
  \XFILE{#1}%
}

% generate and index program name
\newcommand{\PROGRAM}[1]
{%
  \texttt{#1}%
  \XPROGRAM{#1}%
}

% index file name
\newcommand{\XFILE}[1]
{%
  \index{#1@\protect\FILE{#1}}%
  \index{file!#1@\protect\FILE{#1}}%
}

% index program name
\newcommand{\XPROGRAM}[1]
{%
  \index{#1@\protect\PROGRAM{#1}}%
  \index{program!#1@\protect\PROGRAM{#1}}%
}
```

These macros appear to be mutually recursive, since `\FILE` and `\XFILE` call each other. However, no infinite loop develops because the expansions happen at different times: `\FILE` generates a reference to `\XFILE`, but the latter's reference to `\FILE` will not be seen until the index entry is actually read from the `.ind` file.

This trick unfortunately does not work with the `showidx` package/style, because it redefines `\index` in such a way that produces immediate macro expansion.

Note particularly that `\XFILE` and `\XPROGRAM` each make multiple index entries, so that files and programs are indexed under their own names, as well as under 'file' and 'program' entries. Neither of them knows how the index entry is actually typeset; that is left to `\FILE` and `\PROGRAM`. Although in this case, both macros set their arguments in a typewriter font, they nevertheless represent different types of index entries, and in some future version of the document, different fonts might be used. This follows the LaTeX model of structured markup: entries that are logically different have different names, even if in some styles, they may have similar appearance in the output.

Because the macros `\FILE` and `\PROGRAM` are used in the index, they will generate unwanted index entries there unless you disable `\index` like this in your manuscript file immediately before the index is typeset:

```
\renewcommand{\index}[1]{}
\printindex
```

Bibliographies are often sadly neglected by indexers; they should not be. Donald Knuth makes a point of indexing all authors cited in his books. This is handy when you want to find what the author has to say about a particular paper that you found while you were skimming the bibliography. In this document, I've followed Knuth's practice by writing things like

```
\cite{Bentley:EPODD-1-1-3}.%
\index{Bentley, Jon L.}%
\index{Kernighan, Brian W.}
```

Knuth even puts in humorous index entries to amuse sharp-eyed readers; look up the index entry for *Derek, Bo* in the TeXbook [10].

For a large document, such indexing is tedious and error-prone, and a much more sophisticated package for automated author/editor indexing has recently been developed; it is described in [2] and is used in a complex technical book [13].

6 Special characters in index entries

Both LaTeX and `makeindex` reserve certain characters for processing actions, and yet you may need those characters in an index entry. This section describes how to get them.

LaTeX makes minimal requirements on the contents of an `\index` argument: its text must contain balanced braces, even if they are backslashed. Otherwise, no other characters, not even the percent character for beginning TeX comments, have any significance. Of course, when the index file is subsequently read by LaTeX, characters have their normal meanings, so you have to arrange for special handling of these ten:

```
&  $  #  %  _   {  }  ^  ~  \
```

You can represent the first seven by prefixing them with a backslash:

```
\&  \$  \#  \%  \_  \{  \}
```

For the remaining three, you need to define suitable control sequences:

```
\newcommand{\Caret}{\char'\^}
\newcommand{\Tilde}{\char'\~}
\newcommand{\Backslash}
           {\texttt{\char'\\}}
```

For LaTeX indexes, `makeindex` attaches significance to five characters: at sign '@', backslash '\', exclamation point '!', quotation mark '`"`', and vertical bar '`|`'. Because `makeindex` is controlled by style files, it is possible to change all of these characters, but for simplicity, let us assume that the standard style is used.

We have already shown the uses of three of these characters. The quotation mark is used as an escape character to cause the following character to be treated as an ordinary character; it will be removed when the `.ind` file entries are generated. Thus, to get any of the characters `@|"!` into the index as literal characters, you must prefix them with a quotation mark.

A backslash prevents a following quotation mark from having any special significance to `makeindex`, but unlike the quotation mark, it is preserved in such a case. This feature is provided because `\"` is a TeX accent primitive. A backslash before any other character has no special significance to `makeindex`.

Here are some examples:

Index call	Index entry
`\index{one"@two}`	`\indexentry{one@two}`
`\index{three"|four}`	`\indexentry{three|four}`
`\index{five"!six}`	`\indexentry{five!six}`
`\index{f""ur}`	`\indexentry{f"ur}`
`\index{f\"ur}`	`\indexentry{f\"ur}`

A more detailed example is taken from the UNIX manual pages for `makeindex`. It shows the entries in the `.idx` file of all printable ASCII characters other than letters and digits, assuming the default TeX index 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}
```

In the actual fields following the at signs, characters that 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=^= (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}
```

Because TeX will not write text to a file with unbalanced braces, even ones prefixed by a backslash, you could not create the entries

```
\indexentry{"{@"\"{ (left brace)}{123}
\indexentry{"}@"\"} (right brace)}{125}
```

by the input

```
\index{"{@"\"{ (left brace)}
\index{"}@"\"} (right brace)}
```

However, you *could* write some other unique strings, such as `LEFT-BRACE` and `RIGHT-BRACE`, and then postprocess the `.idx` file with an editor to change those strings to braces.

Alternatively, you can put add balancing braces hidden inside a TeX conditional that will discard them when the index is typeset:

```
\index{"{@"\"{ (left brace)\iffalse}\fi
       \iffalse}\fi}
```

This will make the sort key a left brace, and the actual text of the index entry will contain a literal left brace. The two conditionals hide matching right braces that will not appear in the output, but are necessary for TeX to correctly identify the text of the index entry.

Unfortunately, this technique is limited. It cannot be used to make a similar index entry for the right brace, because we need to get that character into the sort key of the index entry, but that cannot be done if braces are balanced. Postprocessing of the index file is then the only solution.

It is regrettable that TeX's I/O model carries the restriction that all output strings must have balanced braces, because it prevents a clean solution to this problem.

7 Long index entries

If you have long index entries, you can break them at a space and continue on the next line, since TeX converts a newline to a single space. Do not indent the second line, or put multiple adjacent spaces or a TeX comment in the `\index` argument, because they will be preserved, and may affect the index sorting.

Here is an example. The input

```
\index{one two three}
\index{one
two
three}
\index{one   two   three}
```

produces the index entries

```
\indexentry{one two three}{6}
\indexentry{one two three}{6}
\indexentry{one   two   three}{6}
```

Notice that the line breaks did not make the first two differ, but the additional spaces in the third are preserved.

8 Controlling macro expansion

As mentioned earlier, the argument of `\index` is treated to little more than a brace balance check, and written out directly to the `.idx` file. In particular, any TeX macros present will *not* be expanded. This is convenient, because those macros will be expanded when the `.ind` file is later read, and their premature expansion would clutter the `.idx` file and possibly interfere with sorting.

However, in LaTeX 2.09 and 2e, `\index` is a *fragile command*. If it is used in arguments to other control sequences, characters other than letters, digits, and punctuation characters must be suitably protected. The major problem is control sequences in the argument to `\index`; these must be prefixed by `\protect` to delay their expansion.

Also, if the `\index` entry is generated by a macro, as in the `\X`, `\XFILE`, and `\XPROGRAM` examples above, macro expansion will occur unless `\protect` is used.

Some examples should make this clear. The input

```
\index{gnat}
\index{\TeX{}}
\index{\texttt{typewriter}}
```

produces `.idx` file entries

```
\indexentry{gnat}{1}
\indexentry{\TeX{}}{1}
\indexentry{\texttt{typewriter}}{1}
```

Notice that no macro expansion has occurred. Now consider these indexing requests:

```
\X{gnat}
\X{\TeX{}}
\X{\texttt{typewriter}}
```

These produce macro-expanded entries:

```
\indexentry{gnat}{2}
\indexentry{T\kern -.1667em\lower .5ex
           \hbox {E}\kern -.125emX{}}{2}
\indexentry{{\ptt typewriter}}{2}
```

The solution is to rewrite the index requests as

```
\X{gnat}
\X{\protect\TeX{}}
\X{{\protect\tt typewriter}}
```

which gives the `.idx` file entries

```
\indexentry{gnat}{2}
\indexentry{\TeX {}}{2}
\indexentry{\texttt{typewriter}}{2}
```

with the prefixing `\protect` macros removed.

One troublesome entry is the LaTeX verbatim text request, `\verb+abc+`. You can write

```
\index{\verb+abc+}
```

and get

```
\indexentry{\verb+abc+}{3}
```

However, if you use it in the argument to another macro, such as `\X{\verb+abc+}`, you'll get several nasty TeX errors:

```
! Undefined control sequence.
...
! Argument of \@sverb has an extra }.
...
! Illegal parameter number in definition
  of \@gtempa.
...
! Argument of \@argdef has an extra }.
...
! Paragraph ended before \@argdef was
  complete.
...
! Use of \@newline doesn't match its
  definition.
...
! Forbidden control sequence found while
  scanning text of \write.
```

Fortunately, `\protect` saves the day. Just write

```
\X{\protect\verb+abc+}
```

to get

```
\indexentry{{\verb +abc+}}{3}
```

If the argument to `\verb` is another `\verb`, a bare `\index` works fine

```
\index{\verb=\verb+\TeX+=}
\index{\protect\verb=\verb+\TeX+=}
```

and produces

```
\indexentry{\verb=\verb+\TeX+=}{4}
\indexentry{\protect\verb=\verb+\TeX+=}{4}
```

However, problems arise with `\X`, and you have to resort to protecting everything:

```
\X{\protect\verb=\protect\verb+\protect\TeX+=}
```

Now a new problem surfaces. The output contains

```
\indexentry{{\verb =\verb +\TeX +=}}{4}
```

Notice the extra spaces. The one that follows the first `\verb` do not matter, since spaces after control sequences are always ignored when TeX reads input. However, the ones following the second `\verb` and `\TeX` *do* matter, because they are part of a verbatim sequence, and will be preserved in the output, even though they were not originally present.

In such situations, you have to be clever to fool TeX. Rewrite the entry as

```
\X{{\protect\tt \protect\bs
verb+\protect\bs TeX{}+}}
```

and you will then get

```
\indexentry{\texttt{\bs verb+\bs TeX{}+}}{4}
```

which will be typeset correctly as `\verb+\TeX{}+`.

Notice that the empty brace pair did not require special handling. It is preserved in the output to the `.idx` file from the `\index` macro, and preserved on input from the `.ind` file because there it is verbatim text.

Backslashed braces pose a similar difficulty when passed to embedded `\index` macros. In a top-level `\index` reference, like

```
\index{/M \{ moveto \} def}
```

they produce problem-free entries:

```
\indexentry{/M \{ moveto \} def}{5}
```

However,

```
\X{/M \{ moveto \} def}
```

produces

```
\indexentry{/M \@lb  moveto \@rb  def}{5}
```

which is acceptable to LaTeX, but not to `makeindex`, because of the at signs. Once again, protection is called for:

```
\X{/M \protect\{ moveto \protect\} def}
```

produces

```
\indexentry{{/M \{  moveto \}  def}}{5}
```

However, extra space has been introduced after the backslashed braces, and this may not be acceptable. It does not appear to be possible to prevent TeX from inserting these spaces, so in such cases, it is necessary to edit the output `.idx` file before processing it with `makeindex`. With a batch text editor, such as the UNIX `sed` utility, this job can be automated. Here is an example from the UNIX `Makefile` used for this document. It removes space after control sequences, as well as `\penalty0` and discretionary hyphens. The penalties and discretionary hyphens would otherwise interfere with sorting, and possibly produce duplicate entries.

```
idxtips.ind:    idxtips.idx Makefile
        mv idxtips.idx idxtips.tmp
        sed -e 's=\\{  *=\\{=g' \
            -e 's=\\}  *=\\}=g' \
            -e 's=  *{}={}=g' \
            -e 's=\\-  *==g' \
            -e 's=\\penalty *0==g' \
            -e 's=\(\\[A-Za-z][A-Za-z]*\)  *{=\1{=g' \
                <idxtips.tmp >idxtips.idx
        $(MAKEINDEX) idxtips
```

9 Checking the index

If you followed the instructions in Section 3 about the order of job steps, and were careful to avoid blank space before `\index` calls, you can be certain that the page numbers in the `.ind` file are correct.

However, there are likely to be other errors that you need to correct. Here are some common ones to watch for:

- Indexing the singular and plural forms of an entry:

```
\item gnat, 2, 11, 17--21
\item gnats, 17
```

- Inconsistent multi-level entries:

```
\item gnat, feeding habits, 19
\item gnat, 2, 11, 17--21
  \subitem feeding habits, 27
```

- Inconsistent spacing in index entries:

```
\item gnat, 2, 11, 17--21
  \subitem feeding  habits, 23
  \subitem feeding habits, 27
\item white space, 19
\item whitespace, 17
```

  Watch for tab characters in TeX documents; they will usually surprise you. In GNU Emacs, you can get rid of them by marking the whole buffer (type `C-x h` to run the function `mark-whole-buffer`) and then typing `M-x untabify`.

- Inconsistent font usage:

```
\item gnat,  2, 11, 17--21
  \subitem feeding habits, 23
  \subitem \emph{feeding habits}, 41
```

- Inconsistent punctuation and abbreviations in index entries:

```
\item Adobe Systems, Inc., 21
\item Adobe Systems Inc., 27
\item Adobe Systems Incorporated, 87
```

- Inconsistent use of hyphens:

```
\item Hewlett-Packard, 9
\item Hewlett Packard, 27
```

- Inconsistent use of capitalization (the computer industry is responsible for numerous tradenames and other words with embedded uppercase letters):

```
\item LaserWriter, 17
\item Laserwriter, 8
```

- Inconsistent use of TeX ties:

```
\item Knuth, Donald E., 10
\item Knuth, Donald~E., 19
```

- Ignoring `makeindex` syntax errors recorded in the `.ilg` index log file. The terminal output when you run `makeindex` should begin something like this:

```
This is makeindex, portable version
2.11 [11-Sep-1991].
Scanning input file idxtips.idx
....done
(474 entries accepted, 0 rejected).
Sorting entries.......done
(4583 comparisons).
Generating output file idxtips.ind....
done (303 lines written, 0 warnings).
Output written in idxtips.ind.
```

  If the rejection count or warning count is larger than 0, look at the `.ilg` file to find out why. It will give the line numbers of erroneous entries in the `.idx` file, and you can trace them from there back to the place in your document where they were generated.

- Indexing related entries inconsistently, instead of using cross-referencing:

```
\item Emacs, 17
\item Emacs, \see{GNU Emacs}
\item GNU Emacs, 3, 11, 19--23
```

- Generating the same entry text two different ways that produce the same typeset result:

```
\item \textsc{PostScript}, 17
\item \POSTSCRIPT{}, 21
```

- Getting different entries for the same text because of discretionary hyphens, or `\penalty0` commands for hyphenless line breaks:

```
\item \textsc{PostScript}, 17
\item \textsc{Post\-Script}, 19
\item \texttt{makeindex.tar}, 8
\item \texttt{makeindex.\penalty0tar}, 3
```

  A batch editor can help you to remove these from the `.idx` file; see Section 8 on page 1007.

- Using similar index phrases that result in redundant nearby entries:

```
\item \TeX{}
  \subitem books about, 24
  \subitem books on, 27, 31
```

- Forgetting to use a separate sort key for a name produced by a control sequence, so that entries are sorted incorrectly:

```
\index{\BibTeX}
\index{BibTeX@\BibTeX}
```

- Sometimes forgetting to use `\protect`, so that some entries are macro-expanded, and others are not:

```
\item {\ptt Makefile}, 17
\item \texttt{Makefile}, 11
```

- Spelling errors:

```
\item typeseting, 12
\item typesetting, 23, 27--39
```

Spelling errors look particularly grievous in typeset documents, because the output looks so nice in comparison. You should make a habit of running one, or preferably, several, spelling programs on your LaTeX input files. Particularly with technical documents, there are likely to be many words, including technical terms, phrases in foreign languages, and personal names, that are unrecognized by spelling checkers. If your spelling program allows you to supply a private dictionary, or list of exceptions, you can create such a list from the exceptions produced the first time you run the speller, after repairing all legimate errors.

To automate this job with the UNIX `make` program, add these lines to the `Makefile`:

```
DETEX           = detex -n
SPELL           = spell

.SUFFIXES:
.SUFFIXES: .ser .dvi .ltx .ind .idx \
           .bbl .bib

.ltx.ser:
        -$(DETEX) $< | \
        $(SPELL) +$*.sok >$*.ser
        cat $*.ser
```

and then type something like

```
make idxtips.ser
```

to produce a new list of spelling exceptions in that file, and also display it on the screen. The exception list is kept in the file with extension `.sok`; you can update it from time to time as new correct exceptions are found by the speller. Just remember to keep the exception list sorted alphabetically; the UNIX `spell` utility unfortunately requires that. If you use the Emacs text editor, then the commands `C-x h` (`mark-whole-buffer`) and `M-x sort-lines` will do the job.

References

[1] Nelson H. F. Beebe. LaTeX editing support. This document described a powerful GNU Emacs editor package, freely available at `ftp://ftp.math.utah.edu/pub/tex/bib/emacs.`, March 1992.

[2] Nelson H. F. Beebe. AUTHIDX: An author/editor indexing package. *TUGboat*, 19(1):??, March 1998.

[3] J. L. Bentley and B. W. Kernighan. Tools for printing indexes. *Electronic Publishing—Origination, Dissemination, and Design*, 1(1):3–18, April 1988. CODEN EPODEU. ISSN 0894-3982.

[4] Pehong Chen and Michael A. Harrison. Automating index preparation. Technical Report 87/347, Computer Science Division, University of California, Berkeley, CA, USA, March 1987. This is an expanded version of [5].

[5] Pehong Chen and Michael A. Harrison. Index preparation and processing. *Software—Practice and Experience*, 19(9):897–915, September 1988. The LaTeX text of this paper is included in the `makeindex` software distribution.

[6] Roger Kehr. xîndy: Ein Flexibles Indexierungssystem. Studienarbeit, FB Informatik, TH-Darmstadt, Darmstadt, Germany, November 1995.

[7] Roger Kehr. xîndy by topic: A flexible indexing system. Technical report, Institut für Theoretische Informatik, TH-Darmstadt, Darmstadt, Germany, March 1997. 14 pp. Included with the xîndy software distribution.

[8] Roger Kehr. xîndy manual: A flexible indexing system. Technical report, Institut für Theoretische Informatik, TH-Darmstadt, Darmstadt, Germany, March 1997. 25 pp. Included with the xîndy software distribution.

[9] Roger Kehr. xîndy tutorial: A flexible indexing system. Technical report, Institut für Theoretische Informatik, TH-Darmstadt, Darmstadt, Germany, March 1997. 11 pp. Included with the xîndy software distribution.

[10] Donald E. Knuth. *The TeXbook*, volume A of *Computers and Typesetting*. Addison-Wesley, Reading, MA, USA, 1986. ISBN 0-201-13447-0. ix + 483 pp. LCCN Z253.4.T47 K58 1986.

[11] Leslie Lamport. *LaTeX; A Document Preparation System—User's Guide and Reference Manual*. Addison-Wesley, Reading, MA, USA, 1985. ISBN 0-201-15790-X. xiv + 242 pp. LCCN Z253.4.L38 L35 1986.

[12] Leslie Lamport. *LaTeX: A Document Preparation System: User's Guide and Reference Manual*. Addison-Wesley, Reading, MA, USA, second edition, 1994. ISBN 0-201-52983-1. xvi + 272 pp. LCCN Z253.4.L38L35 1994.

[13] H. G. Othmer, F. R. Adler, M. A. Lewis, and J. C. Dallon, editors. *Case Studies in Mathematical Modeling—Ecology, Physiology, and Cell Biology.* Prentice-Hall, Englewood Cliffs, NJ 07632, USA, 1997. ISBN 0-13-574039-8. viii + 411 pp. LCCN QH541.15.M3C37 1997.

[14] Joachim Schrod. An international version of *makeindex*. *Cahiers GUTenberg*, 10–11:81–90, September 1991.

[15] Joachim Schrod and Gabor Herr. Makeindex version 3.0. Technical report, Technische Hochschule Darmstadt, Darmstadt, Germany, 1991.

Index

◇ Nelson H. F. Beebe
Center for Scientific Computing
University of Utah
Department of Mathematics, 105 JWB
155 S 1400 E RM 233
Salt Lake City, UT 84112-0090
USA
Tel: +1 801 581 5254
FAX: +1 801 581 4148
Internet: `beebe@math.utah.edu`, `beebe@ams.org`, `beebe@ieee.org`