Index Preparation for TEX Related Documents*

David Salomon

11835 Sapota Dr.,
Lakeside, CA 92040, USA
dxs@ms.secs.csun.edu

June 1992

A beta release of the MakeIndex program has recently become available for the Macintosh computer, and I immediately started using it to prepare the indexes of two new books. MakeIndex is easy to use with LATEX but, since I like to work with plain TEX, I have developed all the necessary macros from scratch. They are presented here for the benefit of anyone who wants a professionally looking index.

A few words about the use of MakeIndex are in order. Either LATEX or TEX is used to create a raw index file (the IDX file) with entries such as \indexentry{abc}{24}, \indexentry{xyz}{108}. MakeIndex then reads the file, sorts the items, merges multiple occurences of the same item on the same page, and creates the final, IND, file with entries such as '\item abc, 11, 24, 101' '\item xyz 15, 76, 108'. The user has to define macro \item to typeset an index item in any desired way, and the entire index is then typeset by \input doc.ind. MakeIndex supports sub- and subsub items (indicated by a '!'), and has many other features (see appendix 1).

It is generally agreed that index preparation is a complex job, and that it should not be fully automated. Certain things are best done manually. My original intent was, therefore, to develop simple macros that are easy to read, understand and modify for specific needs, yet can do most of the work. My experience so far indicates that they typically do more than 90% of the work, so only a small part of the total effort of index preparation needs to be done manually.

The macros described here are divided into two groups, those that write index items on the IDX file, and those that read the final, IND file, and typeset the final index. Following the tradition of *The TEXbook* (p. 423), the circumflex is turned into an active character and is used to indicate index items in the source document. A typical example is ^{abc}. The macros have to take into account the following:

1. Some index items are 'silent', they should be written on the IDX file but should not appear in the document. The macros accordingly accept either 1 or 2 arguments of the forms ^{abc} and ^[xyz]{abc} (but not ^{abc}[xyz]). The string xyz is not typeset but becomes part of the index item on the IDX file.
2. Many documents about TEX use the notation |abc| for verbatim listings. The user should thus be allowed to write ^|abc| (but not ^|abc|{..}). In order that they not interfere with the sorting, the vertical bars should be removed from the IDX file, and be reinserted in the IND file.
3. The notation |\abc| presents another special case. The user should be allowed to write ^|\abc|, and both the vertical bars and the '\' should be removed before sorting. However, because of the special way macro \getpar below reads its argument, the removed items cannot simply be reinstated. The solution is to convert such an index item, in the IND file, to the form \bs abc\\. When the IND file is \input, macro \bs adds the '\' and typesets its argument verbatim.
4. In a document about TEX, certain index items are preceded by a '\'. A good example is the item \TeX. This item should be sorted without the '\', to appear among the Ts. The '\' should thus be removed from this item in the IDX file prior to sorting, and should be appended back to the same item on the IND file before typesetting it.
5. Certain characters, such as '$' and '%', have special meaning in TEX. Since an index item may contain such characters, their special meaning should be temporarily suppressed when writing an item on the IDX file.

Point 5 above is handled by the \sanitize macro which changes the catcodes of most special characters to 'other'. Since '^' is used to indicate an index item, the catcodes of ^, ^^K and ^^A are not changed. \sanitize should be expanded before the arguments of the index macros are scanned. The only macros with arguments are \inxA#1 & \inxB[#1]#2. Note below how \begingroup and \endgroup are used to localize the effect of \sanitize.

*Presented at the 9^{e} NTG meeting, June 4, 1992, Amsterdam.

As a result, things such as '`\insert`', '`#52`' and '`52%`' can easily be written on the index file, while '`J\^orgen`' cannot.

Table 1 shows examples of index items. For each item, the way it is specified in the source file is shown, as well as the resulting text in the document, and the way the item is finally typeset in the index.

Source	Typeset in document	Typeset in index
`^[abc]`	abc	abc
`^[xyz]{abc}`	abc	xyzabc
`^[xyz]{ abc}`	abc	xyz abc
`^|abc|`[1]	`abc`	`abc`
`^|\abc|`[1]	`\abc`	`\abc`
`^[|\abc |]{xyz}`	xyz	`\abc` xyz
`^{\abc}`	replacement text of `\abc`	same
`^[|\abc|!xyz]{}`[2]	nothing	`\abc`, xyz[3]

[1] Note that there is no need for braces in this case

[2] Main argument should be empty.

[3] This is a subitem.

Table 1

To handle the different cases above, a small utility program, `IdxInd`, has been written. It should be run on the IDX file before `MakeIndex` reads it, and on the IND file after it is created by `MakeIndex`. The entire process involves the following steps:

- The document is typeset by TEX and file `doc.idx` is prepared.
- `IdxInd` reads file `doc.idx` and creates file `tmp.idx`. This is called the first pass.
- `MakeIndex` is run on `tmp.idx`. It creates the sorted file `tmp.ind`.
- The first and last lines of the IND file are `\begin{index}` and `\end{index}`. They should be removed manually, since they are only used by LATEX.
- `IdxInd` is run on `tmp.ind` and creates file `doc.ind`. This is the second pass.
- File `doc.ind` is `\input` and the indexing macros (the second set below) used to typeset the final index. Any special index items will show up in this step and will require manual intervention. Also, bad page breaks in the index pages will have to be corrected by the user at this stage, either by changing the penalties used in the macros, or by inserting penalties at strategic points in file `doc.ind`.

Table 2 shows how different types of index items appear in the source file and in the four index files involved.

type	in source file	on index file `doc.idx`	`tmp.idx`	`tmp.ind`	`doc.ind`
1	`^[abc]{}`	`abc`	`abc`	`abc`	`abc`
2	`^|abc|`	`|abc|`	`abc*`	`\item abc*,..`	`\item |abc|`
3	`^|\abc|`	`|\abc|`	`abc#`	`\item abc#,..`	`\item \bs abc\\`
3 (with sub item)	`^[|\abc|!xyz]{}`	`|\abc|!xyz`	`abc#!xyz`	`\item abc#,..` `\subitem xyz,..`	`\item \bs abc\\,..` `\subitem xyz,..`
4	`{^\abc}`	`\abc`	`abc&`	`\item abc&..`	`\item \abc`
4 with) sub item)	`^[\abc!xyz]{}`	`\abc!xyz`	`abc&!xyz`	`\item abc&..` `\subitem xyz,..`	`\item \abc` `\subitem xyz,..`

Table 2

Note that one should not have, in the same document, items such as `^[\abc]{}` and `^|\abc|`. They would be changed by pass 1 to `abc&` and `abc#`, respectively, and would therefore be considered different by `MakeIndex`. Instead of `^[\abc]{}` the author should write `^[|\abc|]{}`. It is also permissible to write `^[|\abc||bold]{}` (with three vertical bars). In such a case, `IdxInd` will only remove the first two bars (and the backslash).

The last example in table 2 is of a main item `|\abc|` and a subitem `xyz`. The entire item must be silent (placed in square brackets), and the pair of braces that follows should be empty.

When `IdxInd` removes a pair of vertical bars (or vertical bars and a '\', or just a '\') it stores a special code following the modified item, so that the removed characters can later be restored. In table 2 the codes shown are an '*', a '#' or an '&'. In reality, the program uses ASCII codes 04, 03 and 02, respectively, for this purpose. Those are the codes of control characters, and so don't show up in print.

Here is the first set of macros, those that write the index entries on the IDX file. Note the use of `\string\indexentry`. Using `\noexpand\indexentry` also works, but writes a space following `\indexentry`.

```
\newwrite\inx
\immediate\openout\inx=\jobname.idx

\def\Caret{\ifmmode\def\next{^}\else\let\next=\indexT\fi\next}
\catcode‘\^=\active \let^=\Caret

\def\makeother#1{\catcode‘#1=12\relax}
\def\sanitize{\makeother\ \makeother\\\makeother\$\makeother\&%
\makeother\#\makeother\_\makeother\%\makeother\~\makeother\|}

\def\indexT{\futurelet\new\macX}
\def\macX{\ifx\new[\let\Next=\inxB\else\let\Next=\inxA\fi
\begingroup\sanitize\Next}
\def\inxA#1{#1\finidx{#1}}
\def\inxB[#1]#2{#2\finidx{#1#2}}
\def\finidx#1{\gdef\wridx{\write\inx{\string\indexentry{#1}{\folio}}}%
\endgroup\wridx}
```

The document, with index items, should follow, ending with `\closeout\inx`.

Special index items: Control sequences and items in angle brackets should be handled in a special way. The control sequence `\abc` should be indexed as `^[\abc]{}` (silent). An item such as `<xyz>` should be indexed as `^[<xyz>]{}` (silent) or `^[<xyz>]{}\lr{xyz}` where `\lr` is a macro producing the brackets.

Since in practice we don't want the brackets and backslashes to participate in the sorting, we should move them to the right in the IDX file (thus `^[abc\]{}` `^[xyz><]{}`), run `MakeIndex`, and move them back to the left in the IND file. For most documents, such cases are rare, and can be handled manually. For documents on TEX, a program should be written to do this. For the index of *The TEXbook*, a special code was written on the raw index file (pp. 423–424) to indicate special items. After sorting, a special program was run to interpret the codes and add brackets, backslashes, etc.

The index itself is prepared in double-column format. The macros for double columns appear in *The TEXbook*, pp. 416–417 and should be personalized. Typically, the values of `\hsize` & `\vsize` should be changed, and the `\shipout` in macro `\onepageout` may have to be modified.

The second set of macros appears below. These are the indexing macros, which are deliberately kept simple, to make them easy to read and modify. They should be sufficient for most needs. Note the following:

- An item such as `^[auxiliary spaces|see{ties}]{}` should appear only once in the document (since no page numbers will be listed for it anyway). The macros shown here will work even if `^{auxiliary spaces}` appears several times, but the `see` must appear in the first occurence.
- An item such as `^[auxiliary spaces|seealso{ties}]{}` should, similarly, appear just once in the document, at the last time `auxiliary spaces` is indexed.

```
\newif\ifpagebreak\pagebreaktrue
\def\item{\begingroup\obeylines\getpar%
                \global\pagebreaktrue}
{\obeylines
\gdef\getpar#1
{\par\ifpagebreak\bigbreak\else\nobreak\fi
\hangafter=1 \hangindent=15pt #1\par
\global\leftskip=0pt \endgroup}}
\def\goodfil{\hfil\penalty-9\hfilneg}
              %similar to \filbreak [353]
\def\see#1#2{{\it see }#1} % If there is
                  % more than one \see or
% \seealso per item, manual intervention
% is necessary.
\def\seealso#1#2{#2; \goodfil{\it see
                                  also }#1}
\def\bold#1{{\bf#1}}
\def\indexspace{\par \vskip 10pt plus 5pt %
                          minus 3pt\relax}
\def\subitem{\par\pagebreakfalse%
                     \leftskip=7.5pt\item}
\def\subsubitem{\par\pagebreakfalse%
                      \leftskip=15pt\item}
```

Assuming that the double-column macros (from *The TEXbook*, pp. 416–417) are available, the index can now be produced by:

```
\pageno=<whatever>
\line{\bf\hfil Index\hfil\hfil}\vskip.1in
A quotation can be placed here

\begindoublecolumns
\tolerance=6000 \parindent=0pt
\input mybook.ind
\enddoublecolumns
\bye
```

Exercise: An index is normally divided into 26 groups, each of items starting with the same letter. Modify

`\indexspace` to typeset the letter preceding each group.

Answer: If we can assume that all 26 letter groups are present, this is easy. The modified macro is:

```
\newcount\Letter\Letter='101
\def\indexspace{\par\vskip...%
                      \advance\Letter1%
           {\bf\char\Letter}\vskip...}
```

and an additional `\indexspace` command has to be placed manually in the IND file before the first group. If some groups may be missing, the simplest solution is to add a parameter to `\indexspace`:

```
\def\indexspace#1{\par\vskip...
              {\bf#1}\vskip...}
```

and supply the argument manually. A general solution may use the following idea: Macro `\indexspace` sets a flag (a `\newif` variable) to indicate that a letter should be typeset before the next item. Macro `\item` should test the flag. If the flag is true, `\item` should isolate the first character of its argument, typeset it (with appropriate vertical spacing) and clear the flag.

Appendix 1: MakeIndex

`MakeIndex` expects an IDX input file with entries of the form `\indexentry{entry}{page number}`. The following are the main options that it recognizes:

- A '!' can be used for sub items and subsub items. There are no subsubsub items. The IDX file entry `\indexentry{abc!opq!xyz}{23}` is converted by `MakeIndex` to the following three entries in the IND file:
  `\item abc`
  `\subitem opq`
  `\subsubitem xyz, 23`
  The user should define macros `\item`, `\subitem` and `\subsubitem` to typeset the index in any desired way.
- A vertical bar is used to indicate a command. The IDX file entry
  `\indexentry{abc|bold}{23}`
  is converted by `MakeIndex` to
  `\item abc \bold{23}` on the IND file. Again, the user should define a macro `\bold` to typeset the page number in the desired way. Other common examples are
  `\indexentry{abc|see{xyz}}{23}` and
  `\indexentry{abc|seealso{xyz}}{23}`.
- An '@' is used to specify a print entry that's different from the sort entry. The IDX file entry `\indexentry{eleven@xi}{23}` will result in '`\item xi, 23`' placed among the E's in the IND file.

Appendix 2: IdxInd

This small utility is written in Modula 2 for the Metrowerks PSE compiler on the Macintosh computer, and is freely available from the author. It has two separate parts, pass 1 and pass 2. In pass 1 it reads an IDX file, scans each record for special characters, replaces them with ASCII codes 02, 03 or 04, and writes the record on a new IDX file, to be processed by `MakeIndex`. After `MakeIndex` creates the IND file, pass 2 is run, to restore the special characters.

The program looks for a pair of vertical bars and for a backslash. It distinguishes three cases:

- An IDX file entry contains a pair of vertical bars. This corresponds to a type 2 record in table 2. The bars are removed, and an ASCII 02 in inserted at the back of the entry.
- The entry contains a pair of vertical bars, the first of which is immediately followed by a backslash. This corresponds to type 3 in table 2. The three characters are removed, and an ASCII 03 is inserted.
- A backslash but no vertical bars. This corresponds to type 4 in table 2. Again, the backslash is removed, and an ASCII 04 is inserted.

Pass 2 performs two tasks. The first is to look for the special ASCII characters and restore the vertical bars and/or backslash. The second task is to look for long, multi-line records, and convert them to a single line. An index item that appears on many pages in the document, may end up as a long record, with many page numbers, in the IND file. Such an entry is broken, by `MakeIndex`, into several lines. It may typically look like:

```
\item abc 3, 11, 18, ...
125, 153, 167,...
180, 242, 394, ...
```

Our simple macros expect each line to start with `\item`, `\subitem` or something similar. The three lines in the above example have thus to be united into a single record, and this is done in pass 2 by removing the carriage return characters at the end of the first two lines.