Some tools for making indexes: Part I

Lincoln Durst

Three previous episodes in this series of tutorials appear in earlier issues of *TUGboat* [**10**#3 (November 1989, pages 390–394), **11**#1 (April 1990, pages 62–68), and **11**#4 (November 1990, pages 580–588)]. The present installment is more or less independent of its predecessors; the approach, however, is similar in spirit. In fact, some of the ideas here inspired development of tools described in the earlier pieces.

There are four or so main steps in making an index using TEX as, indeed, there are when one makes one by hand, as in the good old days. These steps are (1) selecting the items that seem to be reasonable candidates for index entries, (2) sorting them into the proper order (alphabetical order for words, numerical order for page numbers) or into something near to the proper order, (3) combining or eliminating duplicates and resolving inconsistencies between entries that may have come from widely separated parts of the book but that turn up near each other after the sort, and (4) arranging to have the index typeset the way an index is supposed to look. Times have not really changed; these steps, listed in the order they must be performed are still, in spite of technical advances, arranged in order of increasing difficulty.

It is an easy matter to mark the terms in the text that are reasonable candidates for the index and cause them to be written into a file in the order in which they appear in the text. Sorting that file is not as simple as sorting the file of macro names for entries in the bibliography since the terms are paired with page numbers and, unless you're careful or cunning, a simple ASCII sort of numerals, may give you things like

$$1 < 11 < 111 < 2 < 21 \cdots$$

(cognoscenti call this "lexicographic order"). After an automatic sort, there will be other problems as well. Any inconsistencies resulting from capitalization vs. noncapitalization, plural forms vs. singular forms, etc., will have to be resolved. And, finally, there is the problem of distinguishing between major entries and subsidiary ones.

It is a matter of taste whether the last question mentioned here should be considered first of all or last of all: The problem is philosophical since it comes down to whether such logical distinctions must be settled before one starts writing (this makes the problem of what is to be arranged or rearranged quite an abstract question) or whether it only makes sense to worry about the problem after the list of items to be organized is at hand. Long-time readers of *TUGboat* may notice here an echo of the variant of the chicken-and-egg dilemma posed by Richard Southall over 6 years ago [**5**#2, November 1984, page 80] about whether it is desirable (or even possible) to create a manuscript before its printed version has been designed.

This installment is devoted to the first two steps discussed above: The creation of the file of reminders and some of the problems encountered in sorting it.

Creation of the file of reminders. Here we consider a simplified version of Knuth's indexing macros for *The TEXbook*, described there in Appendix E.*

Knuth makes provisions for four kinds of items in his index, of which

Arabic, `by`, `\char`, `<dimen>`

are examples. Ordinary mortals may be able to survive with one category of items for an index. It is surely easier to grasp what's happening if we restrict our attention to the first of Knuth's four cases. Once you understand the elementary case, please feel free to proceed on your own if you decide to write a book about TEX.

Knuth's scheme is to mark terms in the text file that he may want to include in the index (*The TEXbook*, pages 415–416, 423–424). One point here is that this marking can be done at the time of writing, or at any time during revision. Knuth refers to these selections as "index reminders", since they will not automatically create the final entry; many will be edited during later stages of index construction. Suppose Gauss is mentioned in the text, then the word "Gauss" should be marked as a reminder, even though the entry in the index corresponding to it may in the end be expanded to

Gauss, Karl Friedrich (1777–1855)...

Knuth's way of marking a term in the text is to write `^{Gauss}` or `^^{Gauss}`: in the former case, if the word "Gauss" is to be printed in the text at that point and, in the latter case, if it is not to be printed in the text although it is being considered for the index. For example, the source file for FIGURE 1 contains the following passage:

* Other ways to make indexes have been discussed in earlier issues of *TUGboat*, beginning with the first issue, October 1980 (**1**#1, Winograd & Paxton, Appendix A, pages [1]–12) and, more recently, in the November 1989 issue (**10**#3, David Salomon, pages 394–400). Readers interested in alternatives to what we consider here may profitably consult those sources.

```
By means of ^{stops} the performer has
within his power a number of combinations
for varying the ^{tone} and
^^{power, dynamic}
dynamic power.
```

In *The TEXbook*, items of the second category, those with two "hats", `^^{...}`, are said to be "silent". (There is a reason, as we shall see below, for putting the silent entry just before the word(s) to which it applies; putting it on a separate line may make reading the text on the screen a little easier because the whole line can then be skipped.)

Knuth gives code that will write the marked items into a file (in the order of their appearance in the text), as well as code that provides the option of listing them in the margin of the page, such as shown in FIGURE 1. Here is a stripped-down version [cf. *The TEXbook*, pages 415, 423]:

```
    %%% index.rem, first fragment %%%
\newinsert\margin \dimen\margin=\maxdimen
\count\margin=0 \skip\margin=0pt
\newif\ifsilent \newwrite\inx
\immediate\openout\inx=\jobname.inx
\def\specialhat{%
   \ifmmode\def\next{^}%
   \else\let\next=\beginxref
   \fi
   \next}
\catcode`\^=\active
\let ^=\specialhat
```

First we have the allocation of an insert to hold the list in the margin. The dimension of the insert is allowed to be as long as the whole page (`\maxdimen`), but it occupies a vertical space of zero length (`\count\margin=0`) so that it takes no text area away from the current page: As we have seen before, TEX can swallow, with no noticeable qualms, some assertions that might seem to be outrageously contradictory. [For more about inserts, which are covered with characteristic brevity in *The TEXbook*, see David Salomon's tutorial in the November 1990 issue of *TUGboat* (**11**#4, pages 588–605).] Next, an `\if`-switch is allocated to distinquish between the one-hat and the two-hat items and then the file to hold these things is allocated and opened. `^` is ultimately made active and `\specialhat` is defined to replace it. It is, of course, necessary to account for the fact that `^` may perfectly well be used in math mode in its usual way, thus `\specialhat` is defined so that for `\ifmmode` true, `\specialhat` is merely `^`, but for `\ifmmode` false, it becomes `\beginxref`, the next item on the agenda [*ibid.*, page 423]:

```
    %%% index.rem, second piece %%%
\def\beginxref{\futurelet\next
   \beginxrefswitch}
\def\beginxrefswitch{%
   \ifx\next\specialhat
      \let\next=\silentref
   \else
      \silentfalse\let\next=\xref
   \fi
   \next}
\def\silentref^{\silenttrue\xref}
```

We step through this fast shuffle gingerly. At this point TEX has just seen one `^` not in math mode and `\next` is now `\beginxref`. The immediate question that must be settled is whether there is another `^` just after the first one. Consider the two cases `^{Gauss}` and `^^{Gauss}`: In the first case, the next token TEX will find is "{" and, in the second case "^", i.e., `\specialhat`. `\futurelet` (cf. *The TEXbook*, page 207) permits TEX to sneak a peek at that token and set `\next` equal to it; then `\beginxrefswitch` is expanded. If a second hat is seen, the macro `\silentref` removes the `^`, sets the switch `\ifsilent` to true, and calls the macro `\xref`; in the other case, `\xref` is called but `\ifsilent` remains false. [The discussion of `\futurelet` on page 207 is certainly terse; readers interested in other examples of its use may wish to consult Stephan v. Bechtolsheim's tutorial in *TUGboat*, **9**#3, December 1988, pages 276ff.] Anyway, what happens here is that the macro `\beginxrefswitch` is expanded first, after which the `\ifsilent` switch is properly set and `\xref` is called.

Now it is time for `\xref`:

```
    %%% index.rem, third part %%%
\def\xref#1{\def\text{#1}%
   \let\next=\text\makexref}
\def\makexref{%
   \ifproofmode\insert\margin{%
         \hbox{\marginfont\text}}%
      \xdef\writeit{\write\inx{%
         \text\space!0\space
         \noexpand\number\pageno.}}%
      \writeit
   \else
      \ifhmode\kern0pt\fi
   \fi
   \ifsilent\ignorespaces\else\next\fi}
```

The first two lines here replace fifteen (much longer) lines of code in the middle of page 424, where the four-way branch is made to accomodate the four types of index entries Knuth has to juggle. Notice that two copies of the text in the brackets are made by `\xref`, one of them, `\text`, is for the index file and the marginal note, and the other, `\next`, is the item to appear in the text for the case `\silentfalse`. The steps just mentioned are carried out by `\makexref`. Note `\space!0\space` in the definition; this is used to separate the term

for the index from the number of the page on which it appears. The only relevant criterion for selecting a separator for this purpose is that it should be something that will never appear in a candidate for the index. Thus any sufficiently unlikely juxtaposition of characters will do. Knuth uses !0, !1, !2, and !3, for his four cases and our notation is consistent with his. The period after \pageno is there to mark the end of the reminder. If your page numbers include periods (see below), you should terminate the reminder with some other character here and later (e.g., :). Two subtle points here: (1) The \kern0pt suppresses hyphenation when not in proof mode, for consistency with the result in the insert case (cf. Appendix H of *The TEXbook*, pages 454, 455). (2) \ignorespaces suppresses any space, explicit or implied, at the end of "^^{Gauss} ", which means that with, say,

```
$1+i$ is a ^^{Gauss} gaussian prime.
```

or

```
$1+i$ is a
^^{Gauss}
gaussian prime.
```

in the text file, a page break cannot occur between ^^{Gauss} and gaussian: These two items are firmly attached to one another (as if by superglue which, of course, has no stretch).

To complete the construction of FIGURE 1, we have to monkey with the output routine, in order to get the insert into the margin when we are in proof mode. Knuth's version is on page 416; ours is a bit simpler because we're not trying to do all that he must:

```
   %%% index.rem, output routine %%%
\newdimen\pageheight \pageheight=\vsize
\def\onepageout{\shipout
   \vbox{\offinterlineskip
   \makeheadline\pagebody}\advancepageno}
\def\makeheadline{\vbox to 2pc{%
   \ifodd\pageno\rightheadline
   \else\leftheadline\fi\vfill}}
\def\pagebody{\vbox to\pageheight{%
   \ifproofmode\InsertNotes\fi
   \unvbox255}}
\output={\onepageout}
\newdimen\remkern \remkern=-8pc
\def\InsertNotes{\ifvoid\margin\else
   \rlap{\kern\remkern\vbox to 0pt
      {\box\margin\vss}}\fi}
```

This writes over the \output defined in plain.tex, replacing it by \onepageout; here \offinterlineskip pushes the two boxes (\makeheadline and \pagebody) together by eliminating the space between them, and \vbox255 is the box that holds a full page of text. \InsertNotes puts in the contents of the insert \margin when there are any. Finally (more or less) we need fonts for the reminders in the margin:

```
   %%% index.rem, reminders in margin %%%
\font\remfont=cmtt8 \font\strutfont=cmr9
\newbox\rembox
\setbox\rembox=\hbox{\strutfont )}
\def\remstrut{\vrule height 1\ht\rembox
   depth 1\dp\rembox width 0pt}
\def\marginfont{\remstrut\remfont}
```

Knuth puts his index reminders in the right margin; in our case we choose the left one because the right side is already overworked. Conventional wisdom among professional designers of books and other printed material, based on readability studies, dictates that printed text should contain an average of about ten words per line or fewer. This works out to a maximum line measure (\hsize) for ten point type of about 24 picas, with as many as 30 picas only for twelve point type. It follows that one column of 10pt text on an 8.5″ × 11″ sheet should always have left and right margins whose combined width is close to 4″, i.e., nearly half the width of the sheet; this leaves plenty of room for marginal notes on both sides of the text. See, for example, Southall, *ibid.*, page 86. See also *Designing with type*, by James Craig (New York, Watson-Guptill; London, Pitman; revised edition 1980), page 128. *Typography: How to make it most legible*, by Rolf F. Rehe (Carmel, Indiana, Design Research International; fifth edition, 1984) has an extensive bibliography of the research reports of the relevant legibility studies.*

As usual, we have something for prepare.tex:

```
\newif\ifIndRemMark \IndRemMarkfalse
\def\IndexRemindersMarked{\IndRemMarktrue}
\newif\ifproofmode \proofmodefalse
\def\WriteIndexReminders{\proofmodetrue}
```

and something for compose.tex:

```
\ifIndRemMark\input index.rem\fi
```

There are two more options here for the driver file:

```
\IndexRemindersMarked
\WriteIndexReminders
```

The first option inputs index.rem in the composition run, in order to cope with all the \specialhats, and the second writes the reminders into the file and displays them in the margin at the time the index is to be prepared just prior to the final composition run. (prepare.tex, compose.tex, and driver files

* Can it be possible that there are graduate schools in the U. S. whose policies require that theses be presented in a form intended to make them difficult, tedious, or impossible to read? Anti-dissemination, surely!

```
Pianoforte
strings, 2, 3, or 4 strings
case, harp shape
case, wing shape
pianoforte, grand
harpsichord
strings
stops
tone
power, dynamic
strings
struck, by tangents
tangents, See struck
struck, by hammers
hammers, See struck
strings, plucked
quill
strings, twanging of
```

HARPSICHORD, HARPSICON, DOUBLE VIRGINALS (Fr. *clavecin*; Ger. *Clavicymbel, Kiel-Flügel*; Ital. *arpicordo, cembalo, clavicembalo, gravecembalo*; Dutch, *clavisinbal*), a large keyboard instrument (see PIANOFORTE), belonging to the same family as the virginal and spinet, but having 2, 3, or even 4 strings to each note, and a case of the harp or wing shape, afterwards adopted for the grand pianoforte. J. S. Bach's harpsichord, preserved in the museum of the Hochschule für Musik at Charlottenburg, has two manuals and 4 strings to each note, one 16 ft., two 8 ft. and one 4 ft. By means of stops the performer has within his power a number of combinations for varying the tone and dynamic power. In all instruments of the harpsichord family the strings, instead of being struck by tangents as in the clavichord, or by hammers as in the pianoforte, are plucked by means of a quill firmly embedded in the centred tongue of a jack or upright placed on the back end of the key-lever. When the finger depresses a key, the jack is thrown up, and in passing the crow-quill catches the string and twangs it. It is this twanging of the string which pro-

FIGURE 1 A


```
tone, brilliant
tone, incisive
power of expression
accent
pianoforte
harpsichord
orchestra
harpsichord makers
Ruckers
Antwerp, See Ruckers
Antwerp
```

duces the brilliant incisive tone peculiar to the harpsichord family. What these instruments gain in brilliancy of tone, however, they lose in power of expression and accent. The impossibility of commanding any emphasis necessarily created for the harpsichord an individual technique which influenced music composed for it to so great an extent that it cannot be adequately rendered upon the pianoforte.

The harpsichord assumed a position of great importance during the 16th and 17th centuries, more especially in the orchestra, which was under the leadership of the harpsichord player. The most famous of all harpsichord makers, whose names form a guarantee for excellence were the Ruckers, established at Antwerp from the last quarter of the 16th century. (K. S.)

[The writer is Kathleen Schlesinger, editor of *The Portfolio of Musical Archaeology* and author of *The Instruments of the Orchestra*. This text appears in *The Encyclopædia Britannica*, Eleventh Edition, Cambridge, England, 1910, Volume XIII, pages 15, 16.]

FIGURE 1 B

are discussed in the second of the tutorials in this series, cited above.)

A word of warning: In order to keep things simple, avoid all non-letters in reminders, except for ordinary punctuation, or you will have a terrible time when you try to sort the file. Do not use any diacritical marks, ties (~), or control sequences of any kind, and avoid math mode at all costs. These omissions may be corrected when the file for the index is brought to its final form. If necessary, put

```
... described in detail in his
^^{Heiligenstaat Testament}
Hei\-li\-gen\-staat Test\-a\-ment. ...
```

(See also the Postscript, below.)

Sorting the file of reminders. The first thing that must be done to the file of reminders is to sort it in order to get things into alphabetical order. If brevity has blessed you with no page numbers having more than one digit (or if, for some reason, they all happen to have the same number of digits), the problem is simple; use the utility sort that came with your operating system. Otherwise, heavier artillery may be indicated.

Kernighan & Ritchie, in *The C programming language* (second edition), Prentice-Hall 1988, show (section 5.6, pages 108–110) how to sort a file using Hoare's quick sort. If you prefer Shell's sort to Hoare's, see Harbison & Steele, *C: A reference manual* (second edition), Prentice-Hall 1987, pages 210, 211, for the souped-up Knuth-Sedgewick-Harbison-Steele version and incorporate that into the K&R code. It is not hard in either case to modify the code so that the parts of the lines preceding "!0" are sorted alphabetically and the parts following "!0" are sorted numerically. The trick is to change the criterion for swapping to require that two lines are to be interchanged if the alphabetic strings (a) are in the wrong order or (b) they are the same, but the page numbers are in the wrong order. If you are pig-headed enough to insist on page numbers having the form `1-12` [IBM, here `1-2 < 1-12`], `2.17` [Zenith, here `2.2<2.17`], `K-9` [PCTEX], etc., you deserve the consequences, which you might as well interpret as a challenge.*

As hinted above, there's another way to skin the cat, if your page numbers are plain old natural numbers. Fortunately for us, ASCII codes (unlike telephone dials and typewriter keyboards) are set up so that 0 precedes all the other digits. Hence by padding `\pageno` with enough initial zeros to give all page numbers the same number of digits, we can force lexicographic order to coincide with numerical order. You can use the following code, unless you have a thousand pages or more (if you do, you're the only one to be blamed),

```
\def\Pageno{%
   \ifnum\pageno>99
      \noexpand\number\pageno
   \else\ifnum\pageno>9
      0\noexpand\number\pageno
   \else
      00\noexpand\number\pageno
   \fi\fi}
```

and put "`\Pageno`." in the definition of `\makexref` instead of "`\noexpand\number\pageno`.". With this change, you can use the utility sort that came with your operating system, and after you've done the sorting you can arrange to drop the, by then, superfluous zeros.

Postscript. On another occasion, we propose to consider steps one may take to convert the sorted file of reminders into a respectable index. One must expect that some "hand-work" will be necessary, if only to insert middle names, dates, or other details the author considers relevant or important. But there is quite a bit of routine work that may be automated. In addition, with some cunning, it is even possible to overcome the restrictions imposed so far that prohibit diacritical marks and other control sequences within reminders.

Note. The current version of the disk referred to in previous tutorials in this series contains the files `index.rem` and the source for FIGURE 1, in addition to the things mentioned earlier.

It is a pleasure, as always, to acknowledge the help, guidance, and encouragement, of those who have provided any of the above. In addition to all the usual suspects (cited here several times before), I must acknowledge my gratitude to Costa Mylonas, of Brown University and Athens, Greece, who challenged me to produce a collection of flexible tools, preferably simple ones, that might prove useful for authors who want to sit down at their computers and write books. These tutorials are a result of that challenge. I hope that others find them useful or, at the very least, instructive.

◇ Lincoln Durst
46 Walnut Road
Barrington, RI 02806
lkd@math.ams.com

* A lesser challenge is generated by typographical errors in the passages cited at the beginning of this paragraph. In K&R's code, replace `maxlines` by `MAXLINES`; in H&S's explanation, replace "smallest number" by "largest number". (The errors appear in at least the first printings of these two second editions.)