References

[1] Appelt, W. (1988): Typesetting Chess. *TUGboat* 9#3, pp. 284–287.

[2] Rubinstein, Z. (1989): Chess Printing via METAFONT and TEX. *TUGboat* 10#2, pp. 170–172.

◇ Zalman Rubinstein
University of Haifa
Department of Mathematics and Computer Science
Mount Carmel
Haifa 31999 Israel
Bitnet: rsma407@haifauvm

Editor's note: This article uses a new METAFONT chess font produced by Prof. Rubinstein and his colleagues.

Bibliographic Citations; or Variations on the Old Shell Game

Lincoln K. Durst

This is the first of several tutorials designed to introduce users to some of the subtler parts of TEX, to show how to construct tools to make TEX do things you might like to have it do for you, and to encourage you to take off on your own with the construction of other tools you would find useful. These pieces are no substitute for reading *The TEXbook*; in fact they may be considered successful if they get you to study parts of some danger zones you may have been reluctant to wander into before.

We describe ways `plain` TEX may be used to perform various clerical functions, useful for authors of papers or books who choose to do their own TEX coding as they create the "manuscript". There exist excellent, finely-tuned, and versatile systems ready to use "off-the-shelf" made by Michael Spivak ($\mathcal{AMS}$-TEX) and Leslie Lamport (LATEX) which do some of the kind of things we shall be discussing (as well as much more). Newcomers to TEX may find parts of $\mathcal{AMS}$-TEX and LATEX code hard reading, especially if they try to make changes in order to adapt them for their own needs. Our task is not to reinvent the "wheel"; rather it is to explore ideas that may help users understand how some parts of such "wheels" might work. Here we confine our attention to plain techniques which are easily modifiable and can be adapted or improved by users to address situations of special interest to them. The code printed here is given in fragments to illustrate underlying ideas one or a few at a time.

In the first of these columns we consider the question of constructing bibliographies and lists of references in mathematical or other articles or books. The objective is to make TEX do as much of the "clerical" work as possible (or reasonable). In particular, the numbering of items will be automated so that, as revisions are made and material is changed, interpolated, deleted, or shifted around, the citations will be adjusted properly when the text is composed.

There are at least three forms for lists of items cited. Chemists and physicists frequently list items in the order cited, as do historians and others, using superior figures in the text in order to refer to them. In these cases, the lists may appear either as endnotes or as a "list of references cited." In mathematical articles and books, on the other hand, references and bibliographies most commonly are listed in alphabetical order by authors' names, and occasionally in chronological order by date of publication. Mathematicians tend to put citations in the text within square brackets [as parenthetical remarks, like this one], treating them as asides to the reader. A bibliography, in contrast to a list of references, may include items not actually cited. See, for example, *Concrete mathematics* by Graham, Knuth, and Patashnik (Addison-Wesley, 1989).

There are some curious, if not notorious, examples in which items are listed in an apparently random order. See, for example, *Mathematics Magazine*, **61**#5 (December 1988), pages 275–281. This interesting article by Ivan Niven is about what it takes to win at twenty questions when the person giving the answers is allowed to lie. (THEOREM: *One lie is worth five extra questions.*) The bibliography (mislabelled "References") surely deserves an award for innovation.

The Chicago Manual of Style, "thirteenth edition" (University of Chicago Press, 1982), contains, in chapters 15–17, exhaustive discussions of endnotes, bibliographies, etc., and serves as a source of information on the kind of results desired, as well as suggestions for avoiding many problems, some of which no longer exist, especially for users of TEX.

The construction of FIGURE 1 provides an example of one way the desired results may be obtained using `plain.tex`, with the numbering of sections, displays, references, etc., done automatically. Prior to running off final copy, the macros

1. Fermat numbers

\sect.Fermat.

Fermat considered numbers of the form $2^{2^n}+1$, which are now known as the *Fermat numbers*, F_n, and he may or may not have asserted [**3**, pp. 23ff] that he had proved they are primes for all natural numbers n. Subsequently Euler found that the sixth Fermat number,

$$F_5 = 2^{32}+1 = 4294967297,$$

is a multiple of the prime 641. (Early results of this kind will be found in Dickson's history [**2**, volume i] and more recent results in a book by Brillhart, *et al*, published last year [**1**].)

Euler's result for F_5 follows from the elementary facts given in displays 1.1 and 1.2:

$$641 = 5\cdot 2^7 + 1 = 2^4 + 5^4 \qquad (1.1)$$

\disp.Powers.

hence

$$5\cdot 2^7 \equiv -1, \quad 5^4 2^{28} \equiv 1, \quad 5^4 \equiv -2^4, \quad 2^4 2^{28} \equiv -1 \pmod{641}. \qquad (1.2)$$

\disp.Congruences.

I learned this arithmetic trick from Olaf Neumann of Friedrich Schiller Universität, Jena, D.D.R.; he did not tell me who invented it. LKD

2. References

\sect.Refs.

1 Brillhart, John; Lehmer, D. H.; Selfridge, J. L.; Tuckerman, Bryant; Wagstaff, S. S., Jr. *Factorizations of $b^n \pm 1$, $b = 2, 3, 5, 6, 7, 10, 11, 12$ up to high powers*, American Mathematical Society, Providence (Contemporary mathematics 22, second edition), 1988.

\ref.brillhartFOB.

2 Dickson, Leonard Eugene. *History of the theory of numbers*, three volumes, Carnegie Institution of Washington, Washington, D. C. (Publication number 256), 1918, 1920, 1923. (Reprinted by Hafner and Chelsea.)

\ref.dicksonHTN.

3 Edwards, Harold M. *Fermat's last theorem*, Springer-Verlag, New York, Heidelberg, Berlin (Graduate texts in mathematics 50), 1977.

\ref.edwardsFLT.

FIGURE 1.

used may be printed in the margin, as shown, to facilitate making cross references during revision. In this installment we describe how the second section and the first paragraph of the first section were composed in a single pass. In the next installment, we describe how the ideas used here can be supplemented by others to handle forward references to displays, exercises, theorems, sections, chapters, etc., without having to typeset the text twice.

First we construct a separate file containing definitions for items to appear in the bibliography; call it `bibliog.fil`:

```
%%% bibliog.fil %%%
\def\dicksonHTN{...}
\def\edwardsFLT{...}
\def\brillhartFOB{...}
...
\endinput
```

The dots in the definitions represent the text to appear when the bibliography is printed (author name[s], title, publisher, date, etc.). This file could contain other items as well as those required for this occasion, including others related to the current topic (though not actually cited) and any others the author may anticipate wanting to cite in this work or in others on related subjects. Those which prove to be unnecessary can be dropped at the last minute from files to be constructed from this one. The order in which the definitions appear in this file is irrelevant.

We consider first lists of references printed in alphabetical order by author names. If there are not very many items to be cited, it would be easy to construct by hand another file from `bibliog.fil`, say `bibliog.ord`, which contains the lines:

```
%%% bibliog.ord %%%
\bibmac{brillhartFOB}
\bibmac{dicksonHTN}
\bibmac{edwardsFLT}
\endinput
```

sorted into the order in which items are to be printed in the bibliography. For larger cases, this

process can be automated in part in a variety of ways: One may take advantage of keyboard macros or write a program in a high-level language to extract the necessary parts of `bibliog.fil` and perform the required sort. (TEX can be made to do part of this work, as indicated below.) At or near the beginning of the file containing the text to be composed, include the line `\input biblio.prp`, which reads in the following file:

```
      %%% biblio.prp %%%
\newcount\bib   \bib=0
\def\bibmac#1{\advance\bib by 1
   \expandafter
   \xdef\csname #1\endcsname{\the\bib}}
\input bibliog.ord
\def\ref.#1.{{\bf\csname#1\endcsname}}
\endinput
```

What happens when this is TEXed? First a counter, `\bib`, is allocated and initialized. Then, as `bibliog.ord` is read in, new definitions for macros named `\brillhartFOB`, etc., are constructed using `\csname` [see *The TEXbook*, page 40]. These new definitions assign the numbers to be printed in the text at places the bibliographic items are cited, so that, at least temporarily, we have what amounts to `\def\brillhartFOB{1}` and, therefore, `\ref.brillhartFOB.` is just `{\bf 1}`, etc. The citation itself is made in the text by writing, for example,

```
... he may or may not have asserted
   [\ref.edwardsFLT., pp.~23ff]...
```

So far we have had two quite different definitions for the macros `\brillhartFOB`, etc. (first those in `bibliog.fil`, which we haven't really used yet, and now the new ones just constructed). Before we are finished we shall see several definitions for the macro `\bibmac` and other treatments of the bibliographic definitions, some of which will appear in the closing moves of this shell game.

At the place in the text file where the bibliography is to appear, insert the line `\input biblio.set`, which reads in the following file:

```
      %%% biblio.set %%%
\def\bibl#1#2\endbibl{...}
\bib=0
\def\bibmac#1{\advance\bib by 1
   \bibl{\bf\the\bib}%
   {\csname#1\endcsname}\endbibl}
\input bibliog.fil
\input bibliog.ord
\endinput
```

The first line here contains the definition which specifies the shape of the paragraphs in the list of references (e.g., hanging indentation), type size, leading, parskip, etc. Next we reset the counter `\bib` and change the definition of `\bibmac` so that, when `bibliog.ord` is read in again, `\bibmac` actually typesets the bibliography.

For a list of references printed in order of citation, we construct a file to play the rôle of `bibliog.ord` as the text file is being processed by TEX, and we shall require a revised version of `biblio.set`.

Here we must construct a new file, to replace the file `bibliog.ord` used in the previous example, which contains the bibliographic macros listed in the order of their first appearance in the text.

Instead of using `biblio.prp`, we replace it by `citation.prp`:

```
      %%% citation.prp %%%
\newcount\bib   \bib=0
\newcount\Bib   \Bib=0
\newwrite\bibliolist
\immediate\openout\bibliolist=citation.ord
\def\bibmac#1{\advance\Bib by 1
   \expandafter\def\csname
      #1\endcsname{\the\bib}}
\input bibliog.ord
\def\ref.#1.{\expandafter
   \ifnum\csname#1\endcsname=\the\bib
   \ifnum\the\bib<\the\Bib % not done yet
      \advance\bib by 1%
      \immediate\write\bibliolist
         {\noexpand\bibmac{#1}}%
      \expandafter\edef\csname
         #1\endcsname{\the\bib}%
   \fi\fi
   {\bf\csname#1\endcsname}}
\endinput
```

As in `biblio.prp`, we begin by allocating a counter, `\bib`, and set it to 0. This time we allocate another counter as well, `\Bib`, which will count for us the number of items in the list of references and, in addition, we allocate a file into which we shall write things and then open it with the name `citation.ord`. [For information on reading and writing files using TEX, see *The TEXbook*, pages 217–218, 226–228.] Next we redefine `\bibmac` so that it sets every one of the bibliographic macros equal to `\the\bib` (the value in the counter `\bib`) when `bibliog.ord` is read in, which is what happens next. (Instead of `bibliog.ord` one could use the file `bibliog.uns` described below, since neither the order of its lines nor whether it contains items which will not be cited are relevant in this case.) The first subtlety here is in the definition of `\bibmac`, which uses a plain `\def` for the new definitions of `\brillhartFOB`, etc., which are first defined all to be 0. Use of `\def` here, instead of `\edef`, means that each time a reference to one of these macros is encountered and `\bib` is advanced, the value assigned to *each* of the macros is increased by one. What follows next is a procedure that will *fix* the value of the current argument of `\ref.#1.`, while

the others will still be permitted to grow. The definition of `\ref.#1.` does this by using `\csname` again, but this time with an `\edef` instead of `\def`.

If we introduce an `\if`-switch we can combine the two ways for handling references when it is time to print out the list. If the following code is tucked away somewhere near the beginning of things

```
\newif\ifOrdCited   \OrdCitedfalse
\def\RefsInOrderCited{\OrdCitedtrue}
```

we can adopt a more general form of `biblio.set`:

```
   %%% biblio.set %%%
\bib=0
\def\bibmac#1{\advance\bib by 1
   \bibl{\bf\the\bib}%
   {\csname#1\endcsname}\endbibl}
\input bibliog.fil
\ifOrdCited
   \immediate\write\bibliolist
      {\string\endinput}
   \immediate\closeout\bibliolist
   \input citation.ord
\else
   \input bibliog.ord
\fi
\endinput
```

Thus, one may insert the lines

```
[%] \RefsInOrderCited
    \ifOrdCited\input citation.prp
    \else\input biblio.prp\fi
```

following the definition of `\RefsInOrderCited` and either include or not include a percent sign before the first of these three lines to obtain the references in alphabetical order or in the order of first citation, respectively. Here we have a good example of the value of computers and software such as TEX: Authors shouldn't have to fuss over questions at this level of detail, they have more important things to attend to when preparing their ideas for publication. On the whole, it's up to the designers of books and journals to determine the order in which references should be listed, not authors, nor even — for that matter — editors.

There is a practical problem which deserves attention. Suppose typographical errors exist in arguments of `\ref.#1.`; how can they be caught? For this we could exploit a trick discussed by Stephan v. Bechtolsheim in a Tutorial in *TUGboat*, volume 10, number 2 (July 1989), page 205. The idea is that, unless `\csname #1\endcsname` has previously been defined, its value is `\relax`. Using this fact we can test the argument of `\ref.#1.` to see if `\csname #1\endcsname` actually was defined by `\bibmac`; if the test fails, we can arrange to have TEX make a fuss that is not liable to be overlooked. Readers may wish to try their hand at constructing such code. (See the Note at the end of this article.)

At the beginning, we claimed that a short list of references cited could easily be handled manually, as far as constructing the file `bibliog.ord`, given the file `bibliog.fil`. In the general case, there are three or four steps in this task, two of which can be automated more or less satisfactorily. The original file `bibliog.fil` can be converted into one whose entries look like those in `bibliog.ord` and then sorted. If there are entries to be eliminated (items neither cited nor to be printed in the "bibliography"), some judgment should be exercised about when and where to do this. Further judgment may be required to compensate for inadequacies in the choice of strings to name the bibliographic items and in the sorting process employed. The ideal procedure will surely result from a trade-off between manual effort and how much fancy automation one is willing to concoct.

One step is easy, fiendishly simple, using TEX code suggested by Ron Whitney for the purpose:

```
   %%% bibmac.tex %%%
\newwrite\outfile
\immediate\openout\outfile=bibliog.uns
\def\gobble#1{} % TeXbook, p 308, ex 7.10
\def\dropslash{\expandafter\gobble\string}
\def\makebibmac#1#2{\immediate\write\outfile
   {\string\bibmac{\dropslash#1}}}
\let\def=\makebibmac
\input bibliog.fil
\immediate\write\outfile{\string\endinput}
\closeout\outfile
\bye
```

This makes a nice quiz with which to end. Notice that `\gobble` eliminates the following token only, while `\makebibmac` eliminates everything that follows *except for* the very next token. The first argument of `\makebibmac` is therefore the macro being defined in `bibliog.fil`, so `\dropslash` nibbles off the backslash (the first token of `\string#1`) and the rest of it is written into the file `bibliog.uns` wrapped up inside `\bibmac`, which is just what we want.

Next we have to sort the file `bibliog.uns`. If you work with UNIX or MS-DOS, use the command

```
p:sort <bibliog.uns >bibliog.ord
```

Here `p:` represents the path to the file `sort.exe`. If you don't have UNIX or MS-DOS, look up sorting in your system manual, or write a program in some high level language to do the job for you.

The next question is how near are we to our destination at this point in our trip from `bibliog.fil` to `bibliog.ord`? Part of the answer depends on how clever you were naming the bibliographic control sequences. Several options are at hand: Be *very* clever choosing the names; exert much effort devising clever sorting algorithms; spend a little

effort studying the results so far and rearrange by hand any items not yet in proper order. Sooner or later, automated activity must end, and some other kind of thought is indicated.

If you prefer the references listed in chronological order, rather than alphabetical order, you might use macro names like `\BJIIbrillhart`, etc., substituting `A, B, ... I, J` for the digits `0, 1, ... 8, 9` in dates of publication (with more such "digits" at the end to cope with authors having more than one item per year). Then the same sorting process may be used to make the first (rough) sort of `bibliog.uns` as before.

Some authors, especially historians, favor endnotes that are much more extensive than mere bibliographical citations; for endnotes of this kind, some of which may consist of several paragraphs (and may contain cross references to one another), the scheme described above is quite inappropriate. Such endnotes are typographically equivalent to solutions for exercises. How to handle solutions for exercises and discursive endnotes are topics for a later tutorial in this series.

In the next episode, code will be described to produce cross references and marginal notes. In particular, we shall give another version of `biblio.set` containing provision for displaying the marginal notes shown in FIGURE 1.

Note. A disk (5.25 DSDD) containing source text for FIGURE 1 and the code files used to produce FIGURE 1 is available for MS-DOS users who are members of the TEX Users Group. In addition, code is included for trapping typographical errors in the bibliographic citations as well as identifying (for the case of alphabetical order) bibliographic items not actually cited. The disk also contains source text, including the code samples displayed, for draft versions of other tutorials in the pipeline for this series. Send $6 (which includes a royalty for the TEX Users Group) to the address below. Outside North America, add $2 for air postage.

It is a pleasure to acknowledge the generous help and encouragement of Barbara Beeton and Ron Whitney, without which these ideas would not have been developed.

◇ Lincoln K. Durst
46 Walnut Road
Barrington, RI 02806

Macros for Indexing and Table-of-Contents Preparation

David Salomon

Introduction

Two macros are presented and described in detail. The first is very useful for the preparation of an index; the second prepares a table of contents (toc). It should be noted that LATEX has macros for similar purposes. Ours, however, are different. Our index macro can produce both silent and non-silent index items, whereas LATEX's only generates silent ones. Our toc macros can easily be modified by the user to specify any format for the table of contents.

Another important aspect of the macros is that they are described in detail, thereby illustrating the concept of a multi-pass TEX job and the use of several advanced TEX features, such as active characters, file input/output, `\edef`, `\futurelet`, and `\expandafter`.

The Index Macro

A good index is important when writing a textbook. So much so that Knuth, on several occasions (see reference 1 pp. 423–425, and reference 6), said that he does not believe in completely automating the preparation of an index, and he always puts the final touches on his indexes by hand. As a result, "*his books tend to be delayed, but the indexes tend to be pretty good.*"

Index preparation by computer is not a simple problem. References 1–2 discuss certain features that a good index should have, and how to incorporate them in TEX. The macro described here is relatively simple (even though some readers may not think so) and implements only one advanced index feature namely, *silent* index entries. However, as an example of a TEX macro it is very interesting because it illustrates the use of the features mentioned above.

The macro accepts an index item and writes it on a file, for the future generation of an index. Its main feature is the use of *optional parameters.* The macro accepts either one, two, or three parameters, of which only one is mandatory. The main parameter should be delimited, as usual, by braces, and the optional ones, by square brackets '[' ']'. The macro writes all its parameters on the index file, as one string. However, only one parameter, the main one, is typeset. The optional parameters are treated as silent index items, items that should appear in the index but not in the text itself. A good example is a sentence such as: