How to Avoid Writing Long Records to TEX's \write Streams

Peter Breitenlohner

In his article 'Macros for Indexing and Table-of-Contents Preparation' in *TUGboat* 10, no. 3 (1989), pp. 394–400, David Salomon mentions a problem with very long records written to one of TEX's \write streams (p. 399): "... Since each line of the table of contents goes on the file as a record, its size is limited and, as a result we cannot have chapter or section names which are too long. ..."

There seems to be a similar problem in LATEX, since LATEX users frequently request that TEX implementations be able to write very long records (for CMS up to 1024 characters).

There is, in fact, a very simple method to avoid this problem: split the output into several records at all those points where a new line starts in the user's input. The macro \chap defined below (to be used with plain.tex) demonstrates how this can be done. The argument of \chap is typeset in bold face (with line end characters converted to spaces) and written (\immediately) to a file.

```
\newwrite\ind
\def\chap{\bgroup
  \catcode`\^^M\active \chapx}
\begingroup
  \catcode`\^^M\active
  \gdef\chapx#1{%
    \let^^M=\relax%
    \newlinechar`\^^M%
    \immediate\write\ind{%
      \string\ch:#1\string\\}%
    \let^^M=\space%
    \par{\bf#1}\par\egroup}
\endgroup

\immediate\openout\ind=\jobname.ind
\chap{This is
   one (immediate)
   potentially
   very long text}
\immediate\closeout\ind \vfill\break
```

The definition of a similar macro \Chap:

```
\newwrite\inx
\def\Chap{\bgroup
  \catcode`\^^M\active \Chapx}
\begingroup
  \catcode`\^^M=\active
  \gdef\Chapx#1{\bgroup%
    \lccode`\*`\^^M%
    \lowercase{\egroup \def^^M{*}}%
    \edef\x{\string\ch:#1\string\page}%
    \write\inx\expandafter{\x%
      \number\pageno\string\\}%
    \let^^M=\space%
    \par{\bf#1}\par\egroup}
\endgroup
\newlinechar=`\^^M

\openout\inx=\jobname.inx
\Chap{This is
   another (delayed)
   potentially
   very long text}
\closeout\inx \vfill\break
```

demonstrates that things are only slightly more complicated for a delayed \write.

◇ Peter Breitenlohner
Max-Planck-Institut für Physik
München
Bitnet: PEB@DM0MPI11

Tutorials

Forward References and the Ultimate Dirty Trick

Lincoln K. Durst

In this tutorial, we pick up where we left off in the first episode: See *TUGboat* 10, no. 3 (November 1989), pages 390 – 394.

The last page, 401, of Appendix D ("Dirty tricks") of *The TEXbook* describes what surely deserves to be called the ultimate dirty trick. Under the heading "*Syntax checking*" the creator — of TEX, of course! — tells us how to disable the output routine, abolish all fonts, ignore all line and page breaks, and otherwise have TEX race through a file doing nothing more than executing the macros it encounters and can recognize. The original intention for providing this feature, as the name makes clear, was to locate typographical or other errors in the names of macros: Any macro

not recognized will be listed in the transcript of the TEX run. It is a tool that may certainly be used effectively for that purpose. But it has other uses as well.

"Syntax checking" will write what we tell it to write into files that can be read in when the text file is really TEXed. The scheme here is to handle forward references during a "preprocessing" run and postpone typesetting until everything is ready for it to proceed. One may expect that a preprocessing run should take less time than a full TEX run. Just how much difference there may be depends on a number of things; we shall come back to this question later.

Note that preparing indexes, tables of contents or lists of illustrations, figures, or tables does not require two runs, any more than making bibliographic citations does, provided that one is sufficiently indifferent about which sheets may come out of the printer first. Two complete runs are, of course, required if one wants to make forward cross references involving page numbers, such as those found in the telephone companies' *Yellow Pages* ("See our display adv. on page 9,901"), if some of the advertisements do not precede all the listings that refer to them. References to chapters, sections, theorems, figures, tables, or equations ordinarily do not require page numbers: Page numbers in cross references always have been an expensive luxury and, although they might be getting less expensive, they are still a luxury.

We require a file, call it `syntax.chk`, which will do what is described on page 401 of *The TEXbook*. Two realizations of `syntax.chk` have been made by Michael Spivak. One is a subset of `amstex.tex`; the other is in the file `vanilla.sty` distributed by Personal TEX, Inc., as a part of PC TEX. Readers who have access to either of these files should look for `\font\dummy` and copy out everything from its first occurrence through the definition of `\syntax`. Don't forget to put

```
\catcode`\@=11
```

at the top of the file. At the bottom of the file put the following three lines:

```
\syntax
\catcode`\@=12
\endinput
```

Next we prepare what we shall refer to as a "driver file" like the following one for FIGURE 1 (which also accompanied the first installment of this tutorial):

```
   %%% fermat.tex, a driver %%%
   %%% for fermat.src, q.v. %%%
\input prepare.tex
   \ShowMacros
   \RefsInOrderCited
\preprocess{fermat.src}   \bye
\compose{fermat.src}      \bye
```

Here `prepare.tex` is a file, to be described in detail, containing code that controls what goes on in what follows. `\ShowMacros` and `\RefsInOrderCited` are options (the latter was discussed in the tutorial cited above). `\ShowMacros` sets a switch that causes the marginal notes to appear; default is no marginal notes, so it suffices to "comment" this line out (insert % to its left) when final copy is run. The line after the options starts the preprocessing run and terminates the TEX run when that has been completed. For the composition run, comment out the `\preprocess`-line with % at its left and TEX the same file again. The file `fermat.src` is the file containing the text of the piece to be typeset. The macros `\preprocess #1` and `\compose #1` are defined in the file `prepare.tex`:

```
   %%% prepare.tex, first excerpt %%%
\newcount\sectnum   \sectnum=0
\newcount\dispnum   \dispnum=0
\newwrite\macrodefs
\newif\ifShowingMacros
   \ShowingMacrosfalse
\def\ShowMacros{\ShowingMacrostrue}
\newif\ifOrdCited   \OrdCitedfalse
\def\RefsInOrderCited{\OrdCitedtrue}
\newif\ifMakingPages
   \MakingPagesfalse
\def\preprocess #1{%
   \immediate\write16{%
      .....preprocessing.....}%
   \input syntax.chk
   \PrepareDefs
   \immediate\openout
      \macrodefs=\jobname.ref
   \input #1
   \closeout\macrodefs}
\def\compose #1{%
   \immediate\write16{%
      .....making pages.....}%
   \MakingPagestrue
   \ifOrdCited
      \input citation.prp
   \else
      \input biblio.prp
   \fi
   \input compose.tex
   \input\jobname.ref
   \input #1}
```

We interrupt the listing to describe what's here so far. Parts of this file will be recognized as dealing with bibliographic citations as discussed in the previous tutorial.

First, we attend to some preliminaries: Allocations for counters (to number sections and displays), for a file to hold the macro definitions, and for several `\if`-switches.

Next `\preprocess#1` is defined. First it inputs `syntax.chk` in order to cripple plain TEX; what `\PrepareDefs` is and does will be discussed later; a file is opened next that is to contain the macro definitions (in the case of FIGURE 1, the name of the file opened is `fermat.ref`). Finally it TEXs `fermat.src` and closes `fermat.ref`. What is put into the new file, as we shall see, are definitions for the macros whose names are printed in the margin of FIGURE 1.

The third thing here is the definition of `\compose #1`, which begins by resetting the switch for making pages. Next the preliminary work required for handling bibliographic citations is done, as described in the previous tutorial, and finally three files are input. These are: `compose.tex` which contains typesetting information required for setting titles, subtitles, running heads and feet, marginal notes, etc., information not required for the preliminary run; the file `\jobname.ref`, created in the preprocessing run, which contains the definitions used to make the cross references; and finally the file containing the text source is read in again and the typesetting is carried out.

Now for more of `prepare.tex`:

```
%%% prepare.tex, second piece %%%
   % The next pair of definitions
   % will be written over when
```

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

```
      % compose.tex is read in
      % during the composition run:
\def\section[#1//#2]{\Section{#2}}
      % #1 is the name of the section
      % #2 is a "tag" in the macro name
\def\eqnum #1{\Eqnum{#1}}% #1: "tag"
      % The next three definitions
      % are used in both runs
\def\Section #1{\advance\sectnum by 1
   \dispnum=0 % reset in each section
   \edef\ItemNum{\the\sectnum}
   \writedef[Sect//#1//\ItemNum]}
\def\Eqnum #1{\advance\dispnum by 1
   \edef\Itemnum{\ItemNum .\the\dispnum}
   \writedef[Disp//#1//\Itemnum]}
\def\writedef[#1//#2//#3]{%
   \ifMakingPages
      \MakeNote{#1}{#2}%
   \else
      \def\macdef{\expandafter
         \def\csname #1#2\endcsname{#3}}%
      \immediate\write\macrodefs{\macdef}%
   \fi}
\def\sect.#1.{\csname Sect#1\endcsname}
\def\disp.#1.{\csname Disp#1\endcsname}
\def\ref.#1.{\csname\endcsname}
```

First we have the definition of `\section`, whose first argument is the title of the section, the second argument being the "tag" to appear in the name of the macro, for example, "`Refs`" in the name `\sect.Refs.`. The macros `\section` and `\eqnum` have two versions (the second versions are in the file `compose.tex`), one used in the preprocessing run, the other in the composition run. In the first run, all that is done is to write the macro definitions into the file `\jobname.ref`. For `\section` the first argument, which is the title for the section, is discarded on the first pass, since it is not needed for the macro definition. In the composition run, `compose.tex` is read in after `prepare.tex`, so for macros having definitions in both of these files, those in `compose.tex` simply write over the earlier versions given above. The macro `\writedef` on the first run puts the macro definitions into the file `\jobname.ref`, and on the second run creates the notes which may or may not appear in the margin.

The last three definitions listed above in `prepare.tex` deserve some attention. The first two define the macros used for cross references to sections and displays using `\csname`, in such a way that if one of these macros is encountered as a forward reference, it will not cause TEX to stop to report an undefined control sequence. (Recall that `\csname...\endcsname` has the value `\relax` until it has been defined — cf. the previous tutorial.) For the same reason, the third one causes TEX to discard references to bibliographic items, which are of no interest during preprocessing. All the work involved with bibliographies takes place during the composition run.

The macro `\MakeNote`, which appears in `prepare.tex`, is not needed until the second pass; it is defined in `compose.tex`, excerpts of which are considered next.

First we have headings and displays; this part of `compose.tex` writes over definitions contained in the file `prepare.tex`:

```
   %%% compose.tex, first part %%%
   %%%  headings and displays  %%%
      % This file is loaded by \compose
      % after prepare.tex is loaded
% headings
\def\hdingfont{\bf} % plain default (\tenbf)
\newskip\preheadingskip
   \preheadingskip=5pt plus 1pt
\newskip\postheadingskip
   \postheadingskip=5pt minus 1pt
\def\section[#1//#2]{\Section{#2}%
   \nobreak
   \vskip\preheadingskip\centerline
   {\hdingfont\ItemNum.\
      \vadjNote\margtext #1}%
   \vskip\postheadingskip}
% displays
\def\eqnum #1{\Eqnum{#1}\eqno
   (\Itemnum)\eolNote}
```

Here we have the new definition of `\section`; first it retraces the steps made in the earlier run and again calls `\writedef` which, instead of writing to the file, this time calls the macro `\MakeNote` to construct the text for the marginal note. After that it sets the text for the heading. The treatment of `\eqnum` is similar: The source file contains, in display 1.1, `\eqnum{Powers}`. There are at this point three remaining mysteries: `\MakeNote`, `\vadjNote\margtext` and `\eolNote`; these are the control sequences that put the notes (if any) in the margin.

Next, marginal notes:

```
   %%% compose.tex, second part %%%
   %%%      marginal notes      %%%
\font\margfont=cmtt8 \newbox\mbox
\newbox\margbox  \newbox\Margbox
\def\MakeNote #1#2{%
   \ifShowingMacros
      \gdef\margtext{\char`\\%
         \lowercase{#1}#2}
      \global\setbox\mbox=\hbox{%
         \hskip 1pc\vbox to 0pt{\vss
         \noindent\margfont\margtext}}
   \fi}
```

```
\def\eolNote{%
   \ifShowingMacros
      \hfil\rlap{\box\mbox}
   \else
      \null
   \fi}
\newbox\parenbox \setbox\parenbox=\hbox{)}
\def\parenStrut{\vrule height 1\ht\parenbox
   width 0pt depth 1\dp\parenbox}
\def\vadjNote #1{%
   \ifShowingMacros
      \parenStrut\setbox\margbox
         =\hbox{\hskip\hsize\hskip 1pc
            \parenStrut\margfont #1}%
      \setbox\Margbox=\hbox{%
         \raise 1\dp\parenbox\box\margbox}%
      \wd\Margbox=0pt \ht\Margbox=0pt
      \dp\Margbox=0pt \vadjust{\box\Margbox}%
   \else
      \null
   \fi}
```

First a font is chosen and some boxes are allocated to hold material to be printed in the margin. There will be two kinds of boxes in the margin, depending on whether they are called for at the end of a line of printed text or somewhere other than the end. Notice that in `\eqnum` the marginal box is called at the display number (`\eqno`), which is at the right end of the line. This is the case in which the control sequence `\eolNote` is used. It is possible for a section heading to occupy more than one line and it is probable that a bibliographic item will do so. If we want the marginal box in such cases to be even with the number that appears in the first line, we must either know where the first line breaks or have a way to insert the call where the number occurs and arrange for the box to be printed whenever the line does break. In the latter case, we resort to the more complicated control sequence `\vadjNote`. In any event, the definition of `\MakeNote` provides for both cases: the simpler `\mbox` is used in constructing `\eolNote`, and the text itself, `\margtext`, is retained for use in the other case.

The big trick used for notes not inserted at the end of the line is to invoke `\vadjust`, which is described and illustrated on page 105 of *The TEXbook*; `\vadjust` is designed to do just what we need, to wait until the end of the line in which it occurs and when the line ends its argument is put into the vertical list. There is a minor complication caused by the fact that the point in the vertical list at which the insertion is made is located at the depth of the box containing the line with the call to `\vadjust`. That depth is zero if there are no descenders in the text line, but it may not be zero; also there may be descenders in the name of the macro to be printed in the margin and maybe not. In order to make the two baselines even, we resort to a strut, putting an invisible parenthesis in both lines (the text line and the marginal note) and raise the note by an amount equal to the depth of the parenthesis. [By the way, you *could* use `\vadjNote\margtext` with `\eqnum` (put it *before* `\eqno` in the definition), but if you do, the note will appear near the bottom of the display, which may be a good distance below `\eqno` if large fractions, matrices, etc., are involved.] What's more, we hide the note in `\vadjNote` by first putting it into one box (`\margbox`) and then putting that in turn inside another box (`\Margbox`) along with the upward shift by the depth of the parenthesis. In addition, we make the whole thing invisible to TEX by lying about its dimensions, thereby preventing the lines on either side of it from being spread further apart than normal.

The next excerpt from `compose.tex` is pretty dull, by comparison; it contains specifications for the text font and for references:

```
   %%% compose.tex, third part %%%
% Text font
\def\TextFont{} % plain default (\tenrm)
   \TextFont

% References
\newdimen\thehang   \thehang=1.5em
\def\bibfont{} % plain default (\tenrm)
\def\bibstrut{\vrule height 0pt width 0pt
   depth .4\baselineskip}
\def\bibl#1#2\endbibl{\everypar{}\noindent
   \hangindent=\thehang{%
      \bibfont\unskip\vadjNote\margtext
      \hbox to\thehang{%
         \hfil#1\ }#2\bibstrut\par}}
\def\Bibl#1//#2//#3\endBibl
   {\frenchspacing #1 {\it #2\/}, #3}
```

`\Bibl...\endBibl` is used in `bibliog.fil`; for example:

```
\def\brillhartFOB{\Bibl...\endBibl}
```

We leave the task of filling in the dots as an exercise for the reader.

In later tutorials in this series, other material will be added to `prepare.tex` and `compose.tex`; for example, code dealing with exercises and their solutions, discursive endnotes, and other things. In any case, one line is essential: both these files should end with `\endinput`.

Some loose ends: `biblio.set (v. 0.9)` and `\PrepareDefs`. Here is the third version of

biblio.set, which includes provision for printing the names of the macros in the margin:

```
%%% biblio.set (v. 0.9) %%%
\bib=0
\def\bibmac#1{\advance\bib by 1
   \ifShowingMacros
      \xdef\margtext{\char`\\#1}%
   \else
      \let\margtext=\null
   \fi
   \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
```

There is another version (1.0) of biblio.set on the disk mentioned below; it contains code for trapping spelling errors in macro names for bibliographic items, mentioned in the first tutorial.

Finally, how much faster is preprocessing and what about the macro \PrepareDefs, which appears in the file prepare.tex?

Every sufficiently interesting TEX project will, by definition, use some specially constructed macro definitions devised just for it. Some users may wish to keep these in one place for ease of reference; suppose we call that file \jobname.def. For example, if you are working with collections of n-dimensional vectors, you will surely have a number of definitions of the following sort:

```
\def\vect#1{(#1_1,\ldots,#1_n)}
```

in your collection of special definitions. You may have only a handful of such definitions, or you may have hundreds of them, and even if you don't have many, those few may occur hundreds of times in your source text. Here is an example of a judgment call: Should these definitions go into prepare.tex or into compose.tex, or should different versions go into the two files? If there are enough occurrences of such macros in the source text, it might save time during preprocessing if they were all set to be \relax at that stage, with the real definitions postponed until the compositon run, which is where they are actually needed. But if there are not many of them and they don't occur very often, you might as well input \jobname.def during the preparation stage just prior to \input #1.

For whatever reason, if you're impatient (for example) or if a test shows that it will noticeably shorten preprocessing, here is a way to disable the special definitions during that stage. Include another option, \RelaxDefs, in the driver file to control a new \if-switch in prepare.tex:

```
\newif\ifRelDefs   \RelDefsfalse
\def\RelaxDefs{\RelDefstrue}
```

Before we go into how this switch is to be used, let us recognize that \jobname.def may not even exist, that the text requires *nothing* not already provided in plain.tex. (Maybe all we're doing is setting Shakespeare's sonnets, or somebody else's.) So here comes another option, \DefFileExists, which controls the following switch:

```
\newif\ifdeffile \deffilefalse
\def\DefFileExists{\deffiletrue}
```

Then put the following code in prepare.tex:

```
\def\gobble #1{} % TeXbook, p 308, ex 7.10
\def\dropslash{\expandafter\gobble\string}
\gdef\Relax#1#2{\expandafter
   \let\csname\dropslash#1\endcsname
      =\relax}
\def\PrepareDefs{%
   \def\switchdef{%
      \let\Def=\def \let\Font=\font
      \let\Long=\long \let\long=\relax
      \let\def=\Relax \let\font=\Relax
      \catcode`\#=12 \catcode`\^=12 }
   \def\resetdef{\let\def=\Def
      \let\font=\Font \let\long=\Long
      \catcode`\#=6}
   \ifdeffile
      \ifRelDefs
         \switchdef
         \input\jobname.def
         \resetdef
      \else
         \input\jobname.def
      \fi
   \fi}
```

Readers of the first tutorial in this series will recognize the manœuvre here as a variation on the program bibmac.tex described in the construction of bibliog.ord from bibliog.fil. We fool TEX by telling it that # is an 'other' (see *The TEXbook*, page 37) so TEX will not worry when it is found in horizontal mode; similarly for ^, the reason for the latter will appear in a later tutorial (when we consider index construction).

You should also include the following line in compose.tex:

```
\ifdeffile\input\jobname.def\fi
```

so that the definitions will be present when needed.

Here is \jobname.def for FIGURE 1:

```
   %%% fermat.def %%%
% Cf. TeXbook, page 106 "\signed"
\font\smc=cmcsc10
\def\initials #1.{{\unskip\nobreak
   \hfil\penalty50\hskip2em\hbox{}%
   \nobreak\hfil\smc #1\parfillskip=0pt
   \finalhyphendemerits=0\par}}
\endinput
```

This was used to put the initials "LKD" at the end of Section 1 and the caps-small-caps in the caption "FIGURE 1". This is clearly a case in which exercising the option to disable these definitions is not worthwhile. As a matter of fact, \smc is actually seen by TEX in the file fermat.src only when \ifMakingPages is true; and, although \initials is seen in both runs, it's used only once. So why bother? \RelaxDefs is intended to disable the definitions when (a) that actually makes a difference and (b) the job is too big to be worth doing by hand.

According to Knuth (last sentence on page 401, *The TEXbook*), syntax checking, together with writing to files disabled, "usually make TEX run four times as fast." Here we certainly do not want to disable writing to files, and even though we go out of our way to avoid showing TEX things it has no need to see during preprocessing (such as section titles, bibliographic citations, and — in the next tutorial — texts of exercises, solutions, or endnotes), we find that we can achieve speeds in the range one-to-two times as fast.

Last Question: How can you tell whether relaxing some of the definitions is worth the bother? You can time the preprocessing run and see if it makes any difference. A pair of simple programs that may be used to do this will be found on the disk referred to in the Note below; they are written in ANSI/ISO Standard C. The disk contains for both programs the source code and executable versions for MS DOS systems. If you don't want the disk, write the programs yourself: for the first program (start), use difftime() to write into a file the number of seconds since some reference point just before giving the command tex \jobname, and just afterwards, use the second program (elapse) to open the file and subtract its contents from the present number of seconds since the same point of reference. You could use a shell script or ".BAT file" containing, say,

```
start
tex <filename>
elapse
```

to do this. If you don't like C, you can use TEX instead, but if you do, you will have to use minutes instead of seconds. Cf. \time, page 273, *The TEXbook*; to keep things simple, you may wish to avoid using the TEX version in the middle of the night since \time gives you the number of minutes since midnight. The reference point for difftime() used by (at least some) C compilers appears to be New Years 1970.

Note. A disk (5.25in DSDD) containing source text for the figures in these tutorials and the code files used to produce them is available for MS DOS users who are members of the TEX Users Group. The disk includes, in addition to the files mentioned above, source text for this tutorial and some of the others in the pipeline, as well as *TUGboat* style files (described in *TUGboat* 10, no. 3, pages 378–385), that may be used to typeset the tutorials. Send $6 to the address below, which includes a royalty for the TEX Users Group. Outside North America, add $2 for air postage.

I had overlooked the existence of syntax.chk and \font\dummy until Barbara Beeton called them to my attention. Ron Whitney has been a great help when I encountered problems: If I couldn't solve them, he did; and often when I did solve them, he produced better solutions. I am very grateful to them both.

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