Additionally, a real "stress" test was performed with all these implementations of TEX. Text5 was "hardened": `\Lasertrue` (a switch from `TUGboat.sty`) was added (doing real two-column setting in TEX), which usually has the consequence of needing much more TEX main memory. Additionally (in the third degree) all `\eject` commands introduced to use TEX's memory cautiously were removed from Text5.

Only bigemTEX and bigemTEX286 survived this test. The running times were:

- bigemTEX 3:34
- bigemTEX286 3:33

Other implementors are invited to provide copies of their implementations to be run through the same tests, the results to be reported in a future issue of *TUGboat*. I am willing to accept hints and suggestions from the implementors about how to make the tests run as efficiently as possible. I am also willing to send out any files which cause problems and rerun the tests after the problems have been solved.

This test would not be what it is without valuable advice and some test files from Barbara Beeton.

◇ Erich Neuwirth
Institute for Statistics and
Computer Science
University of Vienna
Universitätsstraße 5/9
A-1010 Vienna, Austria
Bitnet: A4422DAB@AWIUNI11

Tutorials

Long-winded Endnotes and Exercises with Hints or Solutions

Lincoln Durst

This is the third in a series of tutorials written for readers interested in exploring some of the subtler areas of TEX. We illustrate the concepts described by considering tools that may prove useful to authors interested in using plain TEX during creation of manuscripts of books or articles. The first and second installments appeared in *TUGboat* 10, no. 3, pp. 390–394, and volume 11, no. 1, pp. 62–68.

In this episode, we consider discursive endnotes and exercises for which hints or solutions are given. We are interested in endnotes and exercises that may be longer than one paragraph and that may cite one another. (We treat the endnote case as a special case of the exercise case.) As in the earlier pieces in this series, one of the main ideas is to get TEX to attend to the numbering of items and to provide marginal notes in working drafts in order to facilitate cross references. Some of the central ideas here have been introduced in the earlier pieces (for example, using TEX to write things into files), but this time we will be forced to devote special attention to some subleties not considered in the earlier cases.

Source code for *The TEXbook* (part of the general TEX distribution; it is described in Appendix E of *The TEXbook*, pages 412–425) reveals that a central idea of Knuth's about exercises and their solutions is that these should not be separated from one another in the text file (*ibid.*, page 422), no matter how far apart they will eventually appear in the finished book (or books, if solutions or answers are to appear in separate units — for example, in a supplementary volume for teachers).

If we keep the exercises and their solutions together in the source file, it will be possible to rearrange their order, add more, or delete some, and let TEX do the renumbering without tampering with the coordination between the exercises and the solutions. Think of this as analogous to the way footnotes are treated in TEX: the text of each footnote is imbedded in the source file at the point at which it is cited. This is also the way complicated endnotes ought to be handled, for exactly the same reasons. And, while we're at it, there's no difficulty in arranging for solutions and exercises to appear next to each other even in draft versions of the printed text until the dust settles and preparation of pages in final form is ready to begin. In addition, some authors may find it helpful to see the text of endnotes in draft versions of the printed text at the places they are called. Code that will make this possible is described below.

In some books, the fifth edition of Harold Davenport's *The higher arithmetic* (Cambridge University Press, 1982) is one, separate sections or appendixes are provided to contain hints for the exercises and their answers. No problem.

Exercises and endnotes, some preliminaries. Eventually we will consider definitions for macros such as

1. Powers, primes, polynomials, and polygons. \sect.pppp.

Exercise 1.1. Show that if n has an odd factor greater than 1, $2^n + 1$ is not a prime number and, therefore, the only primes of this form are Fermat numbers. \exer.oddfact.

Consider a regular polygon of n sides with one vertex at $(1, 0)$ and the others lying on the circle with radius 1 whose center is at the origin $(0, 0)$. If ϕ is the angle between radii from the origin to consecutive vertices, we may write $\epsilon_n^k = \cos k\phi + i \sin k\phi$ for the complex numbers representing the vertices of the polygon; here $0 \le k \le n-1$. These n complex numbers are roots of the polynomial $x^n - 1$, which always has at least the two factors $x-1$ and $x^{n-1}+\cdots+x^2+x+1$ (sum of a geometric series, again!). If n is a prime, it can be proved (see FIGURE 2 for a pair of sources) that the second factor is irreducible. More generally, suppose that ϵ_n is a root of an irreducible polynomial of degree d. It can be proved (same sources) that if p is a prime whose square does not divide n, then d is divisible by $p-1$; and if p is a prime whose square divides n, then d is divisible by $p(p-1)$.

Exercise 1.2. Determine all values of n for which d, the degree of the irreducible polynomial with root ϵ_n, is a power of 2. \exer.power2.

Hints for exercises.

1.1. Consider the sum of the geometric series $1-x+x^2-x^3+\cdots+x^{2k}$. \exer.oddfact.

Answers for exercises.

1.1. First $1 - x + x^2 - x^3 + \cdots + x^{2k} = (x^{2k+1} + 1)/(x + 1)$, so with $n = lm$, $l = 2k+1$, and $x = 2^m$, we have $2^n + 1 = (2^m + 1)(1 - 2^m + 2^{2m} - 2^{3m} + \cdots + 2^{(l-1)m})$. If n has no odd factors greater than 1, it must be a power of 2. \exer.oddfact.

1.2. Suppose p is any prime dividing n. If p^2 divides n, then d has the factor $p(p-1)$ which must be a power of 2, so p is 2; and if p^2 does not divide n, then $p-1$ still divides d which means that $p-1$ is a power of 2. Therefore (cf. Exercise 1.1) all such n have the form \exer.power2.

$$n = 2^m F_{n_1} \cdots F_{n_r},$$

where $m \ge 0$ and all the factors after 2^m, if any, are distinct Fermat primes, F_i (of which only five are known, $0 \le i \le 4$).

FIGURE 1

```
\exercise #1\exer #2\hint #3\answ #4\endexer
\endnote #1\note #2\endendn
```

that will do all the things we want. Parameter #1 provides a "tag" to appear in the name of the macro that will be used to cite the exercise or note (as in the previous tutorial); parameters #2, #3, and #4 represent the texts of the exercise or the note, the hint, and the solution, respectively. While working with the material on the screen, you may find it convenient to begin a new line (indented) for each of the delimiting control sequences, say,

```
    \exercise ⟨tag⟩
    \exer ⟨text of exercise⟩
⟨more of the same⟩
    \hint ⟨text of hint⟩
⟨more hint⟩
    \answ ⟨text of solution⟩
⟨solution, solution, solution⟩
    \endexer
```

or

```
    \endnote ⟨tag⟩
    \note ⟨text of endnote⟩
⟨more of the same⟩
    \endendn
```

This should make it easier to find your way around during revision than it might be if everything were rolled up together into one big wad.

If you're not going to distinguish between hints and solutions or if you're never going to cite any exercises, you can get away with fewer than four

parameters, of course. In our examples for exercises we include all four and leave for the reader omission of any considered superfluous for a given project. In order to provide for the possibility that the text of the endnote or the exercise itself, the hints, or the solutions, may run to more than one paragraph, we use `\long\def`s. For the preprocessing run (see the previous tutorial in this series), we can set (in `prepare.tex`):

```
\long\def
\exercise #1\exer #2\hint #3\answ #4\endexer
   {\Exer{#1}}
\long\def
\endnote #1\note #2\endendn{\Endn{#1}}
```

(Notice that, with these definitions, the texts of exercises, hints, solutions, and endnotes will be discarded during preprocessing. Only the tags for the macro names are of interest at that stage.) In addition to the definitions above, we put the following code in `prepare.tex`, mimicking definitions in the previous tutorial for `\Section` and `\Eqnum` (page 65, column 1):

```
\newcount\exernum \exernum=0
\def\Exer #1{\global\advance\exernum by 1
   \xdef\Exernum{\Itemnum .\the\exernum}%
   \writedef[Exer//#1//Exernum]}
\def\exer.#1.{\csname Exer#1\endcsname}

\newcount\endnnum \endnnum=0
\def\Endn #1{\global\advance\endnnum by 1
   \xdef\Endnnum{\the\endnnum}%
   \writedef[Note//#1//Endnnum]}
\def\note.#1.{\csname Note#1\endcsname}
```

For composition runs in proof mode, hints, solutions, and endnotes could be embedded in the printed text, say on a narrower line measure with smaller type (as in FIGURE 3) in order to distinguish them more easily from the text itself; for this, include the following code in `prepare.tex`:

```
\newif\ifEmbedded \Embeddedfalse
\def\EmbedNotesEtc{\Embeddedtrue
   \ShowMacros}
```

and include `\EmbedNotesEtc` as an option in the driver file. Comment out this option when pages are being prepared for final copy, or for copy fitting in preparation for making final copy, at which time the hints, solutions, and texts of the endnotes should instead be written into the files to be used to typeset them (call these files ⟨*jobname*⟩`.hnt`, ⟨*jobname*⟩`.ans`, ⟨*jobname*⟩`.not`, say).

Endnotes and exercises: First versions. The ideas involved are the same in both cases, but `\endnote` is simpler to work with, so we consider it before `\exercise`. Here is code for `compose.tex`, beginning with the allocation of a file to hold the notes. Notice the extra counter, `\EndNs`. In the special case of endnotes, it serves no useful purpose, since it keeps step with `\endnnum`; for exercises, however, where there will be an option to skip hints or answers, the second counters will function as "pseudo-booleans": zero before the first item in question and positive otherwise. Thus they will be useful for decisions that must be made (a) to open ⟨*jobname*⟩`.not`, `.hnt`, or `.ans` whenever the first endnote, hint, or solution is encountered, and (b) to input the file in question, if it isn't empty, when its contents are to be printed.

```
    %%% Endnotes (simplest form) %%%
\newwrite\endns \newcount\EndNs \EndNs=0
\long\gdef
\endnote #1\note#2\endendn{\Endn{#1}%
   \unskip$^{\Endnnum}$\
      \begingroup\notefont
   \ifEmbedded
      \par{\narrower\parindent=0pt
      \Endnnum.\vadjNote\margtext\
         #2\strut\par}\noindent
   \else
      \global\advance\EndNs by 1
      \ifnum\EndNs=1%
         \immediate\openout
            \endns=\jobname.not
         \immediate\write\endns
            {\topline}%
      \fi
      \ifShowingMacros
         \expandafter\edef
            \csname enmn\Endnnum
            \endcsname{\margtext}%
      \fi
      \immediate\write\endns
         {\Endnnum.\ \noexpand\vadjNote
         {\csname enmn\Endnnum
            \endcsname}}%
      \Write\endns{#2\unskip\newpar}
\fi\endgroup}  % end \notefont
```

Here the first case (`\ifEmbedded`) is the easiest: The note is printed in the text where it is cited and no writing to files is involved (see FIGURE 3). In the `\else`-case, various things occur. For the first note encountered the file is opened and something is written at the top of the file (perhaps only something like `\parindent=0pt` or `\parskip=.2\baselineskip`, etc.; in special cases — see below — there may be other things, as well). Next the macros to be used for cross references are defined and, as in the previous tutorials, there are two cases corresponding to the option of printing macro names in the margins of drafts or not printing them; in both cases the text of the note is written into the file ⟨*jobname*⟩`.not`. (Observe that, for `\ShowingMacrosfalse`, the text of the marginal note is just `\relax`.) For the present, we can define

\Write to be \immediate\write; we will describe another choice later, after we consider some of the risks involved in all this. \newpar may be simply \par to end the note, or it could also provide a \strut or \vskip to create space between adjacent notes.

The subtlest part of the code so far is the method used to send the text of the marginal notes into the file so they will be properly identified when they are read back in. Because the marginal notes are saved for later use, it must be possible to distinguish between them; this problem did not arise in the earlier tutorials because the notes were used immediately after being created. Here we identify these marginal notes by incorporating the endnote numbers in the names of the macros that expand to the texts of the notes. The macros are named \enmn\Endnnum, i.e., \enmn1, \enmn2, etc., which may look strange to any but the more critical readers of page 40 in *The TEXbook*. When using \csname, nonalphabetic characters may appear in macro names: See lines 4 and 5 in the second full paragraph on page 40. Of course, one can never actually write \enmn1 or \enmn2, but can only write \csname enmn\Endnnum\endcsname instead. As a matter of fact these macro names are never used by the author, they are written and seen only by TEX; when expanded they produce the macro names printed in the margin; recall that \margtext is defined in compose.tex as follows:

```
\gdef\margtext{%
   $\backslash$\lowercase{#1}.#2\unskip.}
```

where #1 and #2 are the arguments of \MakeNote (in this case #1 is Note and #2 is the "tag", Source, say, in FIGURE 2). For reasons given in the previous tutorial, the user writes only \note.#1. when making a cross reference to one of the endnotes, here #1 is the "tag" used to identify it. This trick with \csname is what facilitates forward references, as explained last time (page 65, foot of column 1).

\exercise will be handled analogously, the only difference being that there are more components to be juggled. Before going further, we simplify things a bit in order to reduce duplication of code by creating a generic file-writing macro, based on \endnote, for use in the other cases. Let us, therefore, rewrite the definition of \endnote as follows:

```
\long\def
\endnote #1\note#2\endendn{\Endn{#1}%
   \unskip$^{\Endnnum}$\
      \begingroup\notefont
   \ifEmbedded
      \par{\narrower\parindent=0pt
      \Endnnum.\vadjNote\margtext\
         #2\strut\par}\noindent
   \else
      \WriteFile\Endnnum\EndNs\endns %
      {not}{enmn}{#2}%
   \fi\endgroup} % end \notefont
```

where

```
\long\def
\WriteFile#1#2#3#4#5#6{\def\Number{#1}%
   \def\Counter{#2}\def\FileReg{#3}%
   \def\FileExt{#4}\def\Ident{#5}%
   \global\advance\Counter by 1
   \ifnum\Counter=1
      \immediate\openout
         \FileReg=\jobname.\FileExt
      \immediate\write\FileReg
         {\topline}%
   \fi
   \ifShowingMacros
   \expandafter\edef
      \csname\Ident\Number
         \endcsname{\margtext}%
   \fi
   \immediate\write\FileReg
      {\Number.\ \noexpand\vadjNote
         {\csname\Ident\Number
            \endcsname}}%
   \Write\FileReg{#6\unskip\newpar}}
```

With all this in hand, we can write down the definition for \exercise immediately:

```
   %%% Exercises (simplest form) %%%
\newwrite\hints \newwrite\answs
\newcount\Hints \Hints=0
\newcount\Answs \Answs=0
\long\gdef
\exercise #1\exer #2\hint #3\answ #4\endexer
   {\Endn{#1}%
   \smallskip\noindent
     {\bf Exercise \Exernum.}\
      \ifShowingMacros
         \vadjNote\margtext
      \fi
      #2\par\begingroup\answfont
   \long\def\argiii{#3}\long\def\argiv{#4}%
   \ifEmbedded
      \ifx\argiii\relax\else
         {\narrower\parindent=0pt
            Hint:\ #3\par}
      \fi
      \ifx\argiv\relax\else
         {\narrower\parindent=0pt
            Answer:\ #4\par}
      \fi
   \else
      \ifx \argiii\relax\relax\else
         \WriteFile\Exernum\Hints\hints
            {hnt}{exmn}{#3}
      \fi
      \ifx \argiv\relax\relax\else
         \WriteFile\Exernum\Answs\answs
            {ans}{exmn}{#4}
      \fi
   \fi\endgroup}  % end \answfont
```

Three classical construction problems.

We[1] propose to treat of geometrical constructions, and our object will not be so much to find the solution suited to each case as to determine the *possibility* or *impossibility* of a solution.[2]

Three problems, the object of much research in ancient times, will prove to be of special interest. They are

1. *The problem of the duplication of the cube* (also called the *Delian problem*).[3]
2. *The trisection of an arbitrary angle.*[4]
3. *The quadrature of the circle*, *i.e.*, the construction of π.[5]

In all these problems the ancients sought in vain for a solution with straight edge and compasses, and the celebrity of these problems is due chiefly to the fact that their solution seemed to demand the use of appliances of a higher order. In fact, we propose to show that a solution by the use of straight edge and compasses is impossible.

Notes.

1. The text shown here appears on page [1] of Felix Klein's little book *Famous problems of elementary geometry*, based on lectures first given by Klein in Göttingen, Easter vacation, 1894. `\note.Source.`

   The English translation was made originally by W. W. Beman and D. E. Smith (Ginn & Co., 1897), and revised by R. C. Archibald (Stechert, 1930) based on the latter's article in the *American Mathematical Monthly*, volume 21, 1914, pages 247–259. *Famous problems* was later reprinted by Hafner (1950), Dover (1956), and Chelsea (1955, 1962, 1980).

2. The main result used to settle questions concerning constructions possible with straight edge and compasses is the following theorem: *If x, the quantity to be constructed, depends only upon rational expressions and square roots, it is a root of an irreducible equation $\phi(x) = 0$* [with rational coefficients], *whose degree is always a power of* 2. See page [5] of the book cited in Note 1. `\note.Solution.`

3. The irreducible polynomial in question is $x^3 - 2$, whose degree is certainly not a power of two, *loc. cit.*, page [13]. `\note.Delian.`

4. In this case, Klein works with the equation $x^3 = \cos\phi + i\sin\phi$ in the complex plane, *loc. cit.*, pages 14, 15. `\note.Trisect.`

   For another approach, see Louis Weisner, *Introduction to the theory of equations*, Macmillan, New York, 1938, pages 159–162. Weisner bases his argument on the irreducibility, over the field of rational functions $R(t)$, of the equation $4x^3 - 3x - t = 0$ satisfied by $t = \cos\theta$, $x = \cos\frac{\theta}{3}$.

5. In 1882, Lindemann proved [*Mathematische Annalen*, volume 20] that π is a transcendental number, *i.e.*, it is not the root of any polynomial with rational coefficients and, therefore, certainly not a root of one with degree a power of 2. (Cf. Klein, *ibid.*, Part II.) `\note.Pi.`

FIGURE 2

When either a hint or an answer is to be omitted, its argument should be `\relax` or, to be on the safe side, `\relax %` (see below).

So much for the simple stuff! The code listed above can actually be used to produce the results shown in the FIGURES 1–3 (defining `\Write` to be `\immediate\write`), but would fail for examples somewhat more complicated. The rest of this discussion is concerned with limitations of the procedure described above and with ways those limitations may be circumvented.

Files read by TEX. In the first place, when TEX writes something into a file, it writes one line at a time, so that the kind of thing we have been discussing can result in some very long lines: Whole

paragraphs, or collections of several paragraphs, if they appear as a single argument, will be strung out into gigantic lines in the files we have been constructing. The trouble with this occurs as TEX tries to read long lines when the files are \input so their contents can be typeset. Some of the lines produced in making the figures for this tutorial (using the code described above) contain seven or eight hundred characters. When lines from a file being \input are read, TEX writes them to a buffer, which, for some implementations, can accomodate only a thousand or so characters and in some cases, even fewer (cf. *buff_size*, in *TEX: The Program*, sections 11 and 30).

For exercises, this problem may not be serious, since hints tend to be terse, and answers (and even solutions!) probably should be kept succinct on grounds unrelated to typographical questions. With endnotes, however, the situation is frequently quite different, especially in fields such as history or philosophy, where such self-restraint may be less common than in some of the sciences.

Questions to examine: (a) What factors contribute to the length of the lines written into files by TEX? (b) How can the lines be broken into smaller pieces? There is more than one answer to each of these questions.

Why are some of these lines so long? Aside from the fact that some authors can be long-winded, any macros in the lines are expanded when TEX writes them into a file. For example, the macro \TeX whose name has only four characters expands into more than fifty characters, as shown on page 66 of *The TEXbook*; and even something as simple as \bf, an umlaut, or a circumflex, can generate a dozen or so characters when expanded. Maybe there is no limit to the number of characters one could obtain by expanding such things. Thus brevity on the part of authors cannot solve the whole problem, although it surely may help to alleviate it somewhat.

Breaking the lines into shorter ones. Two variations on \obeylines (*The TEXbook*, pages 94, 352) can be used to cut the text into pieces corresponding to linebreaks that are present in the source file; one of these methods is simple and relatively straightforward, the other is more intricate and accomplishes more. Both methods use versions of the macros defined above; they differ in two ways, most significantly in the definitions chosen for the macro \Write.

As with \obeylines, it must be possible to recognize line-ends in the input file: This is done by converting the (ordinarily invisible) end-of-line mark, ^^M, to an "active" character.

Two problems require special care when one attempts to diddle with TEX fundamentals as we are about to do. First, it is very important to confine such irregularity with care; hence:

```
\def\endlineactive{\begingroup
   \catcode`\^^M=\active}
```

Under normal circumstances, ^^M is simply replaced by a space (*The TEXbook*, page 351), but the simpler of the methods here (first suggested to me by Ron Whitney) calls for making ^^M perform the function of the TEX primitive \newlinechar. [See Peter Breitenlohner, *TUGboat*, volume 11, number 1, page 62.] So, we can define:

```
{\catcode`\^^M=\active
\def\breaklines{\endlineactive%
   \let^^M=\relax%
   \newlinechar=`\^^M}}%
\let\fixlines=\breaklines
```

Note that, below, the pair \fixlines, \endgroup delimits the region in which ^^M will be permitted its strange behavior.

The other problem that requires special attention is that it is impossible to monkey with TEX's system of categories after the text involved has been seen by TEX. Therefore we shall cut the macros for endnotes and exercises into two pieces, so we can make the switch after TEX sees the tags and the text of the exercise but before TEX sees the material to be put into the files: the text of the notes, hints, and solutions. We therefore define, in compose.tex,

```
\gdef\Endnote #1\note{\def\tag{#1}%
   \fixlines\Note}
\long\def\Note #1\endendn{\endnote\tag %
   \note #1\endendn\endgroup}%end \fixlines
\long\gdef
\Exercise #1\exer #2\hint{\def\tag{#1}
   \long\def\xtext{#2}\fixlines\Hint}
\long\def
\Hint #1\answ #2\endexer{\exercise\tag %
   \exer\xtext\hint #1\answ #2\endexer %
   \endgroup} % end \fixlines
```

In both these cases, we simply define \Write to be \immediate\write. What we get in the files, then, is a series of lines each of which stands between a pair of end-of-line characters in the arguments containing the notes, hints, and answers. Two observations are appropriate in connection with this first method: (a) All macros will still be expanded (one of the lines produced this way in one of the figures still has over 150 characters in this case), (b) If any line contains a comment (%...), it does *not* contain an end-of-line character so it will be

run in with its successor (with a space between them only if a real space precedes the percent sign terminating the first of the pair of lines; spaces after control sequences do not qualify, of course). Moreover, because macros are still expanded, the file may well contain an at-sign, @, which turns up in the expansions of quite a few so-called "private control sequences" (see *The TEXbook*, pages 344–364); hence, in this case (as well as in the original case, where no lines are broken), `\topline` should contain `\catcode`\string\@=11`, as a safety measure.

The first version (just described) is easy to explain, takes a little care in its use, and will be sufficient in many situations. The second version, to be described next, is considerably trickier (the tricks are all in the definition of `\Write`), and it does much more. In particular, because it does not expand macros, it not only produces files whose lines are even shorter, but the results are more easily read by ordinary humans. It is also useful in a number of situations other than the one that concerns us here. The central idea is described in an article by Ron Whitney in this issue of *TUGboat* ("Sanitizing control sequences under `\write`", p. 620). Next we apply simplified versions of some of the ideas in that article to our problem (simplified because, for endnotes and answers, we need not worry about the number of the page on which the note is cited or the exercise is stated). When one makes a table of contents, for example, a more complicated version will be required, as discussed in the article just mentioned.

Breaking lines without expanding macros. The big trick described in Ron Whitney's report involves using the TEX primitive `\meaning`. `\meaning` takes a token as its argument and, in the case where that token is a defined control sequence, produces a string of "characters" which appear in the argument's expansion. Thus `\meaning\cos` expands to

```
macro:->\mathop{\rm cos}\nolimits
```

What follows the "arrow" here (`:->`) *looks like* the expansion of `\cos`, but — as explained in the sanitizing article — it is not *really* the expansion because all the characters shown above (except for the space) have category 12 (other: none of the above or below) instead of what they ought to have (backslash should really have category 0, left and right curly braces categories 1 and 2, and the letters category 11; see *The TEXbook*, page 37).

The trick is to create a new macro whose (first-level) expansion is what you want to write into the file, apply `\meaning` to the new one, then catch the sequence following the arrow in `\meaning`'s expansion, and write *that* (which will not be expanded) into the file as a string of individual characters. Later, when the file is read, TEX won't remember how those characters got there, it will just take them for what they appear to be. Here's one way to pull this off: First define

```
\def\getMeaning#1:->#2\endget{%
   \def\Meaning{#2}}
```

`\getMeaning` may be used as follows: Suppose that `\Line` represents one of the lines in a paragraph to be sent to a file. Then consider

```
\expandafter\getMeaning\meaning\Line\endget
```

Here `\expandafter` causes `\meaning\Line` to expand to

```
macro:->⟨text of the line⟩
```

and `\getMeaning` throws out everything here not after the "arrow". As mentioned above, our code is a bit simpler than Ron's because we can ignore the number of the page on which the exercise appears or the endnote is cited.

So far we know how to avoid expansion of macros; now we have to cut the lines into shorter pieces so we can use this trick. Again we make `^^M` active in order to find the end-of-lines in the file, but this time we use the end-of-lines as terminators of strings of characters whose "meanings" will be written into the proper files one at a time. In order to distinguish the two cases, we choose for convenience

```
\def\returnactive{\begingroup
   \endlineactive}
```

For the line breaking process, we follow the method indicated in the article cited, and use a `\loop`; see *The TEXbook*, page 217, for the garden variety, and sources cited in Ron's article, for the fancy version being used here:

```
\returnactive %
\gdef\Parseline#1^^M#2\endParse{%
   \def\Firstline{#1}%
   \def\Remainder{#2}%
   }
\long\gdef\Write #1#2{%
   \let\FirstLine\empty %
   \def\Remainder{#2^^M}%
   \loop %
      \expandafter %
      \ParseLine\Remainder\endParse %
      \expandafter\getMeaning %
         \meaning\FirstLine\endget %
      \immediate\write#1{\Meaning}%
   \ifx\Remainder\empty\else\repeat %
   }endgroup % end \returnactive
```

When using this method, set

```
\let\fixlines=\returnactive
```

instead of `\breaklines`, as before.

1. Powers, primes, polynomials, and polygons. `\sect.pppp.`

Exercise 1.1. Show that if n has an odd factor greater than 1, $2^n + 1$ is not a prime number and, therefore, the only primes of this form are Fermat numbers. `\exer.oddfact.`

Hint: Consider the sum of the geometric series $1 - x + x^2 - x^3 + \cdots + x^{2k}$.

Answer: First $1 - x + x^2 - x^3 + \cdots + x^{2k} = (x^{2k+1} + 1)/(x + 1)$, so with $n = lm$, $l = 2k + 1$, and $x = 2^m$, we have $2^n + 1 = (2^m + 1)(1 - 2^m + 2^{2m} - 2^{3m} + \cdots + 2^{(l-1)m})$. If n has no odd factors greater than 1, it must be a power of 2.

Consider a regular polygon of n sides with one vertex at $(1, 0)$ and the others lying on the circle with radius 1 whose

---

Three classical construction problems.

We[1]

1. The text shown here appears on page [1] of Felix Klein's little book *Famous problems of elementary geometry*, based on lectures first given by Klein in Göttingen, Easter vacation, 1894. `\note.source.`

The English translation was made originally by W. W. Beman and D. E. Smith (Ginn & Co., 1897), and revised by R. C. Archibald (Stechert, 1930) based on the latter's article in the *American Mathematical Monthly*, volume 21, 1914, pages 247–259. *Famous problems* was later reprinted by Hafner (1950), Dover (1956), and Chelsea (1955, 1962, 1980).

propose to treat of geometrical constructions, and our object will not be so much to find the solution suited to each case as

FIGURE 3

Even with the fancier method there are some details to note. For example, if `^^M` is inside a group (such as `{\it ...^^M...\/}`, as in the theorem quoted in note 2, FIGURE 2), it will be passed over when the loop breaks the lines. To force the lines to break in this case, use

```
\it ...^^M...\/\rm
```

instead, say, or even

```
\bgroup\it ...^^M...\/\egroup
```

Failure to correct this problem has two consequences: you will write some long lines into the file and the `^^M`s within the groups will no longer be invisible. Thus, when you later input the file, you will lose everything after the first `^^M` between the two curly braces, up to and including the second brace and anything between that and the next `^^M`. In this situation the visible `^^M` is interpreted as a line-terminator and functions as a comment character.

In both of the cases in which `^^M` is active, it is essential to replace `\relax` by `\relax %` or the test made to determine that a hint or a solution should be omitted will fail: in these cases the argument will be `\relax ^^M` unless the end-of-line has been commented out. As Ron Whitney puts it so picturesquely, it can be difficult to induce one part of the anatomy, even TEX's, to simulate some other part and be "wholly successful" in the process.

The following code brings together the three cases we have been describing. It provides, in `compare.tex` the working definitions for `\fixlines`, `\topline`, and `\Write` for the three cases considered above.

```
\ifEmbedded
   \let\fixlines\begingroup
\else
   \gdef\newpar{\par\vskip 2pt}
   \ifexpmacros
      \let\fixlines\breaklines
      \gdef\topline{\parindent=0pt
         \catcode`\string\@=11}
      \long\gdef\Write{\immediate\write}
   \else
      \let\fixlines\returnactive
      \gdef\topline{\parindent=0pt}
      \returnactive %(*)
      \gdef\Parseline#1^^M#2\endParse{%
         \def\Firstline{#1}%
         \def\Remainder{#2}}%
      \long\gdef\Write #1#2{%
         \let\FirstLine\empty%
         \def\Remainder{#2^^M}%
```

```
            \loop %
               \expandafter %
               \ParseLine\Remainder\endParse %
               \expandafter\getMeaning %
                  \meaning\FirstLine\endget %
               \immediate\write#1{\Meaning}%
               \ifx\Remainder\empty\else%
            \repeat}%
         \endgroup % end \returnactive %(*)
      \fi %
   \fi %
```

Notice that `\returnactive` must be used not only when the lines are actually broken, but also for the definitions of `\Parseline` and `\Write` since they involve the active `^^M`. The definition to be used here for `\loop` is that in Ron Whitney's paper (note the `\else\repeat`). The switch `\ifexpmacros` is controlled by the following code in `prepare.tex` for yet another option, `\ExpandMacros`:

```
\newif\ifexpmacros \expmacrosfalse
\def\ExpandMacros{\expmacrostrue}
```

The swarm of %-signs is here to prevent the overactive `^^M`s from creating mischief. In a more advanced course, you may learn a simpler way to tame them (cf. the "sanitizing" paper in this issue).

Special challenge. Consider the endnotes in Edith Hamilton's *The Greek way* (W. W. Norton, 1942): No marks appear in the text itself, but each note indicates the page number and the line number on that page to which the note refers. Had TEX existed in 1942, how might this have been achieved?

In the next tutorial, we consider some questions related to the construction of indexes. Among other ideas, there will be more about parsing by context and some examples of other ways to use loops.

Note. A disk (`5.25in` DSDD) containing source text for the figures in this series of four tutorials, and the code files used to produce them, is available for MS DOS users who are members of the TEX Users Group. Send $6 (which includes a royalty for the TEX Users Group) to the address below. Outside North America, add $2 for air postage.

As usual, Ron Whitney has been generous with ideas and inspiration. He has taught me a lot about TEX — everything I know about it, except for all the things I learned from Barbara Beeton, over a period of several years, and things that I understood when I read about them for the first time in *The TEXbook*.

◇ Lincoln Durst
46 Walnut Road
Barrington, RI 02806

Output Routines: Examples and Techniques. Part III: Insertions

David Salomon

Note: Before reading this article, the reader should glance at parts I or II for disclaimers and remarks on notation.

Insertions are considered one of the most complex topics in TEX. Many users master topics such as tokens, file I/O, macros, and even OTRs before they dare tackle insertions. The reason is that insertions **are** complex, and *The TEXbook*, while covering all the relevant material, is somewhat cryptic regarding insertions, and lacks simple examples. The main discussion of insertions takes place on [115–125], where TEX's registers are also discussed. Examples of insertions are shown, mostly without explanations, on [363–364, 423–424]. There is, therefore, a need for an article like the present one. It tries to explain insertions in detail, and shows specific, simple examples. Concepts are developed gradually, and the ultimate truth revealed in steps.

Introduction

Definition: An *insertion* is a piece of a document that is generated at a certain point but should appear in the document at another point.

Common examples of insertions are footnotes, endnotes (Note 1), and floating insertions. These are important features, which explains why a general insertion mechanism has been incorporated into TEX. The following short quote (from [124]) says it all: "*This algorithm is admittedly complicated, but no simpler mechanism seems to do nearly as much.*" Using insertions, it is possible to accumulate material (text/pictures) in a box and typeset it anywhere in the document. The material can be inserted on the current page, it may be *held over* by TEX and inserted on the following page, it may be *split* between the current page and the next one, or it may wait for the end of the document. The `plain` format also provides very convenient macros, based on the general insertion mechanism, to handle footnotes and floating insertions.

A good example of insertions is the placement of index items in the right margin [423–424], an operation that is part of the *manmac* format [App. E]. See (Note 2) for an outline of the idea. A simple version is developed elsewhere in this article.

It is important to point out that, even though the insertion mechanism of TEX is general and complex, it cannot deal with every conceivable situation. Consider the case of *facing figures* (Note 3).