Getting \answers in TEX

Jim Hefferon

The tutorial *Long-winded Endnotes and Exercises with Hints or Solutions* by Lincoln Durst (*TUGboat* 11, no. 9, pp. 580–588) gives a clear explanation of why setting up a format like

```
\begin{exercises}
    ⋮
  \item ...
    \answer{-text-}
    ⋮
\end{exercises}
```

is hard. The problem is that writing *-text-* to an auxiliary file is more complex than it seems. Briefly, TEX grabs the argument to `\answer`, expands all tokens fully (looking for things like page numbers that must be evaluated now), and then writes the result as a single—possibly very long—line to that auxiliary file.

Besides being hard to edit, those long lines may choke TEX's input mechanism since *buf_size* may be only a thousand characters.

Long-winded ... details a fascinating trick to try to work around this problem. But for me this approach has two problems. First, I don't entirely understand it (so if it breaks I can't fix it). Second, it won't take line returns inside of braces.

Just as I don't understand the macro, I don't entirely understand the macro's problems, so I am reluctant to rely on it. This note is to point out that an obvious kludge has some advantages.

I defined `\answer` to write the number of the current exercise to the file `exnos.tex`:

```
\newwrite\ansnos
\immediate\openout\ansnos=exnos
\outer\long\def\answer#1{%
  \immediate\write\ansnos{\thelabeledtext} }
```

(`labeledtext` counts the formal parts of my document). A typical output file looks as follows.

```
1.2.1.3
1.2.1.4
1.2.2.7
  ⋮
1.2.2.13
```

Then I wrote an editor macro to find instances of `answer{-text-}`, match them with lines from `exnos.tex`, and write the result to `answers.tex`. After executing the macro, `answers.tex` looks like Figure 1. (The lines of the answers are, comments and all, as I typed them.)

My editor macro, listed in the Appendix, is written in REXX (specifically Personal Rexx for the Kedit editor on an IBM PC).

This solution may involve leaving TEX but it has the advantages that I can modify it to suit my needs, and that it has no obscure restrictions. Stretching TEX is fun, but maybe it just isn't right for this job.

Figure 1. `answers.tex`

```
\par\noindent{\bf 1.2.1.3}
\par{Known as Fermat's Last Theorem,  % !!index this
this result is easy when
\( n \) is infinite or \( 2 \), but is harder
for intermediate \( n \).}

\par\noindent{\bf 1.2.1.4}
   ...
```

Appendix

```
/* ANSWER.kex   THIS IS A REXX PROGRAM            */
/*  Used to separate answers from the file in TeX */
/*  Expects the counters for those answers to be  */
/*  in the file texauxfile, one to a line.        */
/*  Kills .w and .z.                              */
/*   jim hefferon St. Mike's College. 91-1-21     */
answerfilename='answers.tex'
texauxfile='exnos.tex'
```

```
prefixstring='\par\noindent{\bf '
postfixstring='         }\par\nobreak '

/* trace ?i */
'set msgmode off'  /* Matches the "on" below. */

/* Remember the filename and where we are. */
'extract /fileid/'
'set point .z'
'top'
'/\answer{/'           /* find string like this */
failedtofind=rc
'set point .w'         /* .w is last place found answer */

/* Go to output file. Add header (announce the new page). */
'k 'answerfilename
'bot'
'sos addline'
'text %ANSWERS!!'

/* Enter main loop.  */
do while failedtofind=0
  'k 'answerfilename        /* Open the answerfile */
  'bot'                     /* and insert any pre- */
  'sos addline'             /* number info.        */
  'text '||prefixstring     /*                     */
  'k 'texauxfile              /* Open the counter    */
  'down'                      /* file and take the   */
  'reset block'               /* next line.          */
  'mark line'                 /*                     */
  'k 'answerfilename        /* Reopen the answer   */
  'bot'                     /* file, copy next line*/
  'copy block'              /* and put in post-    */
  'sos addline'             /* number info         */
  'text '||postfixstring    /*                     */
  'k 'fileid.1                          /* Go to main (TeX)    */
  'locate .w'                           /* file and stream     */
  'extract /curline/'                   /* mark the answer,    */
  parse value curline.3 with s1 '\answer{' rest  /* e.g.,    */
  'extract /line/'                      /* \answer{Hello, this */
  'cursor file 'line.1 length(s1'\answer{') /*  ~~~~~~~~~~~~ */
  'mark stream'                         /* is it.} More stuff. */
  'cmatch'                              /* ~~~~~~~             */
  'sos makecurr'                        /*                     */
  'mark stream'                         /*                     */
  'k 'answerfilename            /* Transfer to answer  */
  'bot'                         /* file.               */
  'sos addline'                 /*                     */
  'copy block'                  /*                     */
  'k 'fileid.1                    /* Any more answers?   */
  'reset block'                   /*                     */
  '/\answer{/'                    /*                     */
  failedtofind=rc                 /*                     */
  'set point .w'                  /*                     */
```

```
 end

/* Cleanup. */
'k 'texauxfile
'qquit'

/* File the answer file. */
'k 'answerfilename
'file '
/* Move back to where we were when the macro was called. */
'k 'fileid.1
'set point .w off'
'locate .z'
'set point .z off'
'set msgmode on'   /* Matches the "off" above. */
'msg Answers appended to '||answerfilename||'.'
```

◇ Jim Hefferon
Mathematics
St. Michael's College
Colchester, VT 05439
BITnet: **hefferon@smcvax**

Oral TEX

Victor Eijkhout

TEX knows two sorts of activity: those actions that can be classified under 'execution', and those that fall under 'expansion'. The first class comprises everything that gives a typeset result, or that alters the internal state of TEX. Examples of this are control sequences such as `\vskip`, macro definitions, and all assignments.

Expansion activities are those that are performed by what is called the mouth of TEX. The most obvious example is macro expansion, but the command `\the` and evaluation of conditionals are also examples. The full list can be found on pages 212–215 of the TEXbook [1].

In this article I will give two examples of complicated macros that function completely by expansion. Some fancy macro argument delimiting occurs, and there are lots of applications of various conditionals. For a better understanding of these I will start off with a short section on the expansion of conditionals.

About conditionals

For many purposes one may picture TEX's conditionals as functioning like conditionals in any other programming language. Every once in a while, however, it becomes apparent that TEX is a macro processor, absorbing a stream of tokens, and that conditionals consist of nothing more than just that: tokens.

Consider the following example:

```
\def\bold#1{{\bf #1}}
\def\slant#1{{\sl #1}}
\ifsomething \bold \else
             \slant \fi {word}
```

If the 'something' condition is true, the whole `\if ... \else ... \fi {word}` sequence is *not* replaced by `\bold {word}`; instead TEX will start processing the 'true' part of the conditional. It expands the `\bold` macro, and gives it the first token in the stream as argument. Thus the argument taken will be `\else`. TEX will only make a mental note that when it first encounters – more precisely: expands – an `\else` it will skip everything up to and including the first `\fi`[1].

[1] The reader may enjoy figuring out why in spite of the apparent accident in this example the 'word' will still be bold, and why TEX will report that 'end occurred inside a group at level 1' at the end of the job.