SOME TEX PROGRAMMING HACKS

Leslie Lamport
SRI International

When writing a complicated TEX macro, one is essentially writing a program. Since TEX is a macro substitution language, writing programs in it can be a bit tricky. This note describes how to implement some ordinary programming language features in TEX. This allows you to write a macro by first writing it as an ordinary program, then translating it into TEX. I find this to be quite helpful.

For program control structures – in particular, while and if statements – I refer you to Brendan D. McKay's macros that appeared in TUGBOAT. I will concentrate on assignment statements and data structures. First, let me define some terms.

text: Any TEX input containing balanced braces.

variable: A TEX macro name, beginning with \ – *e.g.*, \foo. I will use \var as a generic variable.

val(\var): A piece of text that is the value of the variable \var.

eval(*text*): The result of evaluating *text* by recursively replacing all nonprimitive macro names and all \counter expressions by their values. For example, if \count5 = 13, *val*(\foo) = \bar yz and *val*(\bar) = xx, then

$$eval(\text{\hbox to \count5 pt\{\foo\}}) = \text{\hbox to 13pt\{xxyz\}} \quad .$$

Note that *eval* is idempotent, so

$$eval(eval(text)) = eval(text) \quad .$$

The first thing to note is that TEX provides a block-structured language, so each variable can nested definitions as in Algol. In TEX, a block is begun by a { and ended by a }. This means that there are two kinds of assignments:

\var := ...: Changes the the value of \var in the local block.

\var :== ...: Changes the the value of \var in the local block and all enclosing blocks.

Here are some kinds of assignment statements that you can construct.

\var := *text*. It is implemented in TEX as:

\def\var{*text*}

Note that if *text* consists of a single variable name, this makes \var synonymous with that variable.

\var :≡ *text*. It is implemented as:

$$\gdef\var\{text\}$$

\var :≡ *eval*(*text*). Implemented as:

$$\xdef\var\{text\} \quad .$$

\var :≡ *val*(\varx), where \varx is another variable. It is implemented as:

$$\let\var=\varx \quad .$$

There are two data structures that I have figured out how to implement: a push-down stack and an array. The macros are the following.

\vpush\stk{*text*} – pushes *eval*(*text*) onto stack \stk.

\vpop\var\stk – performs the assignment

$$\var :\equiv head(\stk)$$

and pops stack \stk.

\rdarray\var\array{*i*} – performs the assignment

$$\var := \array(eval(i)) \quad ,$$

where *i* should evaluate to an integer.

\vrarray\array{*i*}{*text*} performs the assignment

$$\array(eval(i)) \equiv text$$

For these array operations, \array must be defined to have the form

$$\downdownarrows\var_1\downdownarrows\var_2\ldots\downdownarrows\var_n$$

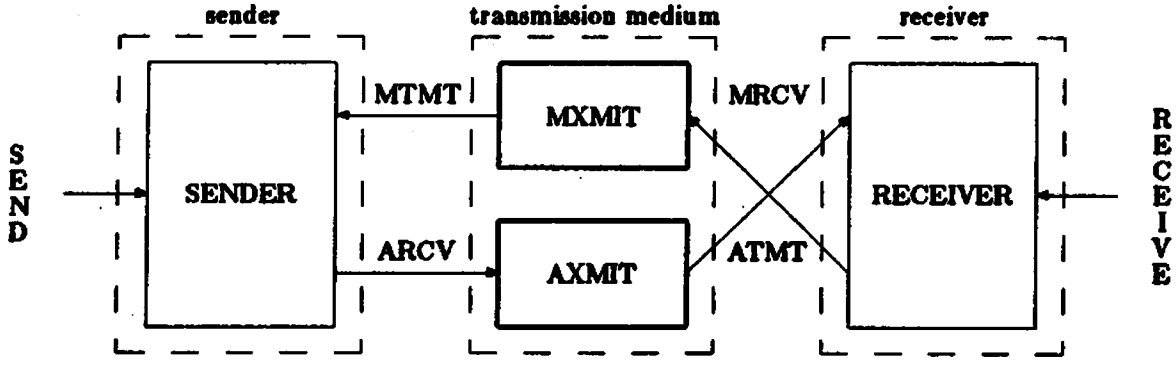
where the $\var_i$ are variables not used elsewhere.

These macros are defined as follows.

```
\def\vpush#1#2{\xdef #1{\xdef \ii{#2}\xdef #1{#1}}}
\def\vpop#1#2{#2\xdef #1{\ii}}
\def\rdarray#1#2#3{\def\ii##1{\ifpos9{\gdef\tmp{\let#1=##1}\advcount9by-1}\else
                    {}}\setcount9#3#2\tmp}
\def\wrarray#1#2#3{\def\ii##1{\advcount9by-1\ifzero9{\gdef##1{#3}}\else
                    {}}\setcount9#2#1}
```

. . . . . .

Editor's note: The preceding item is reproduced from Canon copy supplied by the author. He provided the editor with several other documents which were not able to be processed satisfactorily in time for the deadline (the editor's fault, not Les'). Les is using a DEC 10-compatible system, and has developed a macro package (FaCSL TEX,based on Max Díaz' Fácil TEX) with an EMACS preprocessor (PRETEX) capable of performing syntax checks and expanding bibliographic citations from references stored separately from the root file. One particularly interesting feature permits a user to create "just about any kind of picture you want that doesn't contain curved lines", such as:



This feature requires fonts not yet available at AMS. We will try to install them and present the details in the next issue.

* * * * * * * * * * *

UNBLOCKING AN $\mathcal{A}\mathcal{M}\mathcal{S}$-TEX TAPE

Barbara Beeton
American Math Society

Several recipients of $\mathcal{A}\mathcal{M}\mathcal{S}$-TEX tapes written for computers other than DEC 10/20s have complained that the tape format does not conform to the description supplied with the tape. The description specifies fixed records, fixed blocks, with the implication that no carriage return or line-feed codes are present. In fact, these tapes contain variable-length records, blocked, with each record terminated by a CR/LF.

Donald C. Wells, of the National Radio Astronomy Observatory, was kind enough to send a listing of a VAX/VMS Fortran program which "reads entire blocks, of arbitrary length, and scans them character by character to build up the proper lines of text. [The] program produces an auxiliary file which tabulates the lengths of all the blocks in the 20 files on the tape. ... It might be of assistance to someone else who is using a VAX under VMS."