Warnings & Limitations

Controlling <ctrl-M>; Ruling the Depths

Barbara Beeton

Two problems have recently surfaced to snare some unwary users. One (the easy one) has to do with how TEX handles the depths of rules in the absence of explicit instructions. The other, a more insidious problem, is what happens when TEX encounters a "bare" control-M in its input stream. Let's take the nasty one first.

Beware of the bare <ctrl-M>

We received a phone call in which it was pointed out that some sections of text were missing from a published paper for which the author had created the input and the camera copy was prepared at AMS, An investigation uncovered the fact that in some lines of text, a "bare" carriage return, or <ctrl-M>, occurred in the middle, in fact, immediately before the missing text. This was the condition of the file when we received it, and the file had not been changed subsequently. However, nothing had been missing on the proof, which had been prepared by the author.

Something had happened that is not supposed to happen with TEX: the same file, run through two different implementations of TEX, had produced different results.

We demonstrated the system-specific nature of this problem by constructing the following test file and running it through three different implementations of TEX available to us:

```
The quick brown^Mfox jumped over the
lazy dog.
```

On a DEC-20/TOPS-20, using the Stanford implementation, the output read "The quick brown fox jumped over the lazy dog." On a VAX/VMS, both the Stanford and the Kellerman & Smith implementations yielded "The quick brown lazy dog."

While talking with a colleague who uses a VAX/Unix Pastel implementation of TEX, I mentioned the problem and he ran the test through his system. This yielded a third result: an error message,

```
! Text line contains an invalid character.
```

The TEXbook, page 343, declares that `\catcode'\^^M=5`, which on page 37 is interpreted as "end of line". On the systems with which I am most familiar (DEC-20/TOPS-20 and VAX/VMS), the traditional end-of-line marker is an ASCII carriage-return/line-feed pair (⟨*crlf*⟩ or `^M^J`). In files I have received from Mactexintosh users, the bare `^M` seems to be the norm. (TEXing an unaltered Mac file on a DEC-20 invariably exceeds the input buffer, so I had grown accustomed to translating Mac `^M`s to ⟨*crlf*⟩s on receipt of a file, if the author had not already made the conversion.) And I understand that some other systems prefer record-oriented input, with no explicit end-of-line marker.

It is thus apparent that there are several interpretations to the bare `^M`, and they seem to be operating system dependent, or at least implementation dependent.

- The Stanford TOPS-20 implementation ends a line and assumes that the character which follows starts a new one; in other words, `^M` by itself, although nonstandard, is equivalent to ⟨*crlf*⟩.
- The Stanford and K&S VAX/VMS implementations end consideration of the content of the line (treating the `^M` the same as a `%`) and skip to the next ⟨*crlf*⟩.
- The Unix Pastel implementation does not permit a bare `^M`, only a ⟨*crlf*⟩ pair.
- There may be others — this investigation has really only just begun.

I inquired about this varied behavior, and was informed by David Fuchs that it was really a user problem, attributable to incorrect transfer of the file from one system to another. He said,

> Different run-time I/O systems for different compilers work differently, and there's nothing we can do about it. ... TEX is line-oriented; when moving TEX files around, you must choose a technique that preserves lines in the respective native modes of the various machines you are moving between. On various different systems this may mean mapping between CR, CRLF, LF, or padding out to a fixed record length (80, for instance) while dropping any CR/LF, or even putting a byte-count at the start of a variable-length record that may or may not need a CR/LF. Whatever is the native mode of the system is exactly what is correct; if you just carelessly transfer bits, you're out of luck!

Asking for further clarification from Don Knuth, I received the following reply.

> This is not a bug in TEX, since different installations do support different character

> sets. There are two ways to make a TEX file machine dependent, both mentioned in the manual: (1) Name your files with local conventions that use something other than letters and digits. (2) Use non-printing ASCII characters in your source files.
> ...
> Some installations of TEX will accept TAB characters (and treat them as spaces); others will not. ... See *TEX: The Program*, section 23; the *xchr* and *xord* can be different on different computers.
> ...
> In other words, TEX is working just as we designed it. The users who expect identical behavior across machines have to abide by (1) and (2).

I'm still not happy, because there is still a problem. More and more TEX users are shipping files around to other sites, where they are to be TEXed to generate copy for publication, or read, perhaps changed, and shipped on somewhere else. Not all these locations will have identical hardware and software, and, given the opportunity, Murphy's Law always applies.

I don't have any good solution to this problem. The only (bad) solution I can think of is, if your system is one that handles this situation poorly, you should check every file for bare `^Ms` and fix them up before submitting the file to TEX. I would welcome a discussion of this problem by the TEX implementors, since, although it appears to be a communications problem between disparate systems, it does seem to bend the "same input, same output" principle that we've come to expect TEX to follow.

Rules can be a deep subject

One of my correspondents sent me an interesting puzzle a while ago. In what he thought was a simple use of a "fill-in" rule in an `\halign`, he suddenly found that the rules varied in thickness. Here is the preamble he was using:

```
\halign{# \hfill
   &# \vrule width 12pc height .5pt \cr
... }
```

and this is what he got:

Name: _____
Address: _____
_____
Telephone: _____
Best time to call _____

The rule on the next-to-last line has taken on the depth of the "p".

This is what he wanted:

Name: _____
Address: _____
_____
Telephone: _____
Best time to call _____

I can see two possible solutions.

- Explicitly specify the depth; otherwise the "environment" depth will be used. In the "corrected" version above, `depth 0pt` was added.
- Use leaders instead of a `\vrule`: `\leaders\hrule\hskip 15pc` gives the same result as the `\vrule`.

Actually, my correspondent had tried an `\hrule` first, but got the nasty message that leaders were required in that context. Remembering that one must always nest opposites (`\vrule`s in horizontal mode, `\hrule`s in vertical), and being too lazy to look up how to use leaders, he simply switched to the `\vrule`. A reasonable approach. But there are a few things that one must remember not to take for granted.

Macros

German TEX, a Next Step

Peter Breitenlohner

1 The Present State

As reported by Joachim Lammarsch (TUGboat 8(1987)304) and described in detail by Hubert Partl (TUGboat 9(1988)70–72) the German TEX Users Group has agreed on a standard for a "Minimal Subset of German TEX Commands" at its 6th meeting in Münster (Germany) last October. This standard was, in fact, designed to a large extent by Hubert Partl and the present implementation in terms of TEX–macros is almost entirely his work.

The basic idea is old: make " an active character and define " as a macro with one argument such that the macro expansion yields whatever is necessary.