A puzzling TEX macro

Peter Schmitt

What would you answer if I ask you to define a control sequence whose replacement text is a single left brace `{`? You would answer (probably without hesitation) that such a control sequence does not exist. How would you react if I insist, and still ask you to generate such a control sequence? You probably will produce the TEXbook, and point to a passage which explicitly states that this is impossible, or you might suspect some trick question and try to find a double meaning in my question.

But I am stubborn and repeat my challenge more precisely:

1 Problem

Define a macro which can be used to define a control sequence which expands to a single left brace `{`.

Or, more precisely:

After the application of the macro, say `\MakeBrace\brace`, the replacement text of `\brace` should consist of a single token, a character of category 1.

The following remarks should disturb all doubts about hidden meanings in the formulation of the 'puzzle':

(1) If the problem is correctly solved and if `\cs` is a macro that takes a single (undelimited) argument,
then `\expandafter\cs \brace` ⟨*argument*⟩`}`
is equivalent to `\cs {`⟨*argument*⟩`}`

(2) If `\def\Def{\def\cs}`
then
`\expandafter\Def \brace` ⟨*macro text*⟩`}`
is equivalent to `\def\cs {`⟨*macro text*⟩`}`

Moreover, I concede the following passage:

(3) According to the TEXbook (page 203), for macros "all occurrences of `{` and `}` in the ⟨*replacement text*⟩ are properly nested."

And I stress:

(4) Of course, `\def\MakeBrace {{\iffalse}\fi}` is not a solution of the problem.

Are you intrigued? Do you want to try to find out by yourself? Then be warned: It is *not* a trick question, but the solution *is* tricky, and it involves a *trick* you might consider to be unfair. (Hint: Obviously, re-reading the TEXbook is *not* likely to help you!) For all the others who are just curious and only want to check if (or where) I am cheating — here is how to do it:

2 The Solution

The solution of the problem is based on an observation (over which I stumbled quite accidentally) concerning the behaviour of the `\read` command. `\read` stores "the contents of the next line" in a control sequence `\cs`, but "additional lines are read, if necessary, until an equal number of left and right braces has been found." If the end of the file is prematurely reached, TEX complains and issues an error message (`File ended within read`). However, quite surprisingly, the input read so far is available in the control sequence `\cs` which therefore contains an unmatched left brace `{`.

The definition of `\MakeBrace` takes advantage of this behaviour. First it generates a file containing a single brace `{` only. (Of course, this file could also be prepared manually.) Then the file is opened for input and (using `\read to#1`) it is read to the control sequence to be defined. Before that, the `\endlinechar` is set to `-1`, i.e., it is removed, since otherwise an additional `\par` would be read. Moreover, in order to hide the error message, the macro temporarily switches to `\batchmode`.

```
\def\MakeBrace #1{\bgroup \batchmode
%       % supress error message
  \immediate\openout1 brace
    \escapechar-1
    \immediate\write1{\string\{}
%       % write a single {
    \immediate\closeout1
  \immediate\openin1 brace
    \endlinechar-1 \global\read1 to #1
%       % (incomplete) read of {
    \immediate\closein1
                \egroup \errorstopmode }
%       % return to normal
% test:
\MakeBrace\brace  \show\brace
\message{.\brace.}}
%                % note second } !
```

3 Remark

Attempting to apply the same idea to a right brace fails: When `\read` encounters an unmatched brace `}` TEX does not even bother to stop but (silently!) discards the rest of the line, including the offending brace. Consequently, only properly nested braces are read, and the unmatched brace has the same effect as a comment character.

4 Summary

Reading (by a `\read` command) from a file which is not balanced with respect to `{` and `}` causes TEX to behave in a surprising way which is not documented in the TEXbook. In one case, an unmatched `}` is not reported, in the other case, an unmatched `{` produces a macro which — according to the rules — cannot exist. Does this constitute a bug or a feature of TEX?

◇ Peter Schmitt
Institut für Mathematik
Universität Wien
Strudlhofgasse 4
A-1090 Wien, Austria
schmitt@awirap.bitnet
schmitt@pap.univie.ac.at