Marcin Woliński

Multiple expansions triggered with a single \expandafter

This pearl (coded on October 18, 1996) is the most useless one I could think of. Nonetheless it is an example of a really curious expansion of macros.

Let us imagine that we have a list of non-space tokens and we want to assign this list to a token register without expanding the tokens and in reversed order. Here is a simple macro that reverses a list in an expand-only way:

```
\def\afterfi#1#2\fi{\fi#1}

\def\reverse#1{\reverseX{}#1\stopreverse}
\def\stopreverse{\noexpand\stopreverse}

\def\reverseX#1#2{\ifx\stopreverse#2%
    \afterfi{#1}%
  \else
    \afterfi{\reverseX{#2#1}}%
  \fi}
```

Now we can write

```
\message{\reverse{abcdefg}}
```

and TEX will respond with writing `gfedcba` on the terminal.

To put the result of reversing the list `abc\foo def\bar ghi` in a token register we do the following:

```
\toks0=\expandafter{\if0\reverse{abc\foo def\bar ghi0}}\fi
\showthe\toks0
```

With the use of `\expandafter` we introduce a single expansion to the region where expansion is suppressed. The token being expanded is the `\if`. To expand an `\if` TEX needs to find next two non-expandable tokens to compare them. The first token is `0`, but then TEX sees the macro `\reverse`. So the macro gets expanded. An interesting feature of `\reverse` is that no non-expandable tokens are emitted until the list is fully reversed. So only then TEX stops expansion. The first non-expandable token TEX will see is the second `0`, which we have devilishly inserted at the end of the list. At this point the condition turns out to be true and the next tokens get assigned as contents to the token register.