TEXniques

Victor Eijkhout

The ultimate loop macro

Introduction

The plain TEX format contains a `\loop` macro that has been a source of frustration and puzzlement to users ever since. Its syntax is somewhat strange, you have to insert an `\if...` condition in it but cannot use `\else`, and nested use of the macro runs into various problems. In this article I will describe my own improved loop macro, which I've called `\repeat` to prevent confusion.

You can get the macros from CTAN:

```
http://tug.ctan.org/cgi-bin/CTANfilesearch.pl?FILESTRING=repeat
```

keywords
...

User interface

Looping constructs have been common in programming languages for a long time. My loop macro is vaguely modelled on the Algol68 construct: the syntax is

```
\repeat
   \for{<var>} \from{<start>} \by{<step>}
               \to{<end>} \downto{<end>}
               \until{<cond>} \while{<cond>}
   \do { <loop body> }
```

Some remarks about this:

- All control sequences in between `\repeat` and `\do` are optional; if you leave them *all* out, you get an infinite loop.
- If a 'for' variable is specified, for instance `\for{i}`, a control sequence `\i` is available in the loop body. Strictly speaking, this control sequence has been `\let` to a counter that is allocated by the package. This loop variable can also be used as a bound for any nested loops.
- The loop body is written inside braces, but there is no implied grouping, so all assignments are global.
- The step size is always positive; it is added or subtracted depending on whether `\to` or `\downto` is used. The default is, of course, an increasing counter, stepping by 1.
- The 'until' test is evaluated at the end of the loop body; the 'while' test at the start. The condition is any TEX `\.` test. To terminate the loop with a test somewhere in the middle of the loop body, use

  ```
  \ifsomething ... \expandafter \breakrepeat \fi
  ```

Implementation

Above, I mentioned the fact that the `\repeat` macro can be used nested; in fact, it can be nested to as many levels as you want. Now, I also mentioned that the loop has a counter. So, do I allocate whole bunch of counters to beging with? Nope. Here's the crucial bit:

```
\newcount\REPdepth
\def\repeat#1\do{%
   \advance\REPdepth by 1
   \REPcsargrom\ifx{REPcount}\relax
     \REPcsargrom{\csname newcount\expandafter\endcsname}{REPcount}%
     \REPcsargrom{\csname newtoks\expandafter\endcsname}{REPtoks}%
```

where

```
\def\REPcsargrom#1#2{%
  \expandafter#1\csname#2\romannumeral\REPdepth\endcsname}
```

That is, a new counter is allocated for each level, the first time it is encountered. A unique token list for each level is also allocated to hold the loop body.

The macro goes on:

```
   \fi \REPsetup{#1}%
   \edef\REPtmp
     {\def\REPcsargrom\noexpand{REPrepeat}%
          {\REPcsargrom\noexpand{REPbody}}}%
   \REPtmp
   \afterassignment\REPdxbody\REPcsrom{REPtoks}}
```

where

```
\def\REPcsrom#1{\csname #1\romannumeral\REPdepth\endcsname}
```

The `\REPsetup` call processes all the options, then the `\edef` trickery defines control sequences such as `\REPrepeatii` (on level 2) as `\REPbodyii`; this superfluous looking step is necessary because we terminate the loop by redefining `\REPrepeatii` as `\relax`. The `\afterassignment` sets aside the 'define and execute' macro `\REPdxbody`, and the token list `\REPtoksii` is then assigned whatever comes after `\do` (remember that the argument of `\repeat` was delimited by `\do`?); in other words, the loop body.

The 'define and execute' macro of the loop body goes like this:

```
\def\REPdxbody{%
   \REPcsargrom\edef{REPbody}{%
     ... % the while test
     \noexpand\the\REPcsargrom\noexpand{REPtoks}%
     ... % the until test
     ... % counter update
     \noexpand\endrepeat
     \REPcsargrom\noexpand{REPrepeat}}%
   \REPcsrom{REPbody}}
```

Above we had defined `\REPrepeatii` as `\REPbodyii`, so together this is a clean case of daisy-chain recursion.

Ending the loop is done by, as promised, by defining away the `\REPrepeatii` control sequence:

```
\let\endrepeat\relax
\def\breakrepeat#1\endrepeat{%
   \REPcsargrom\let{REPrepeat}\relax
   \advance\REPdepth by -1\relax
   }
```

Of course, I have left out plenty of detail here, but this should convey the flavour of these macros.

Examples

If you retrieve the file from CTAN, you'll see various examples at the end, after an `\endinput` statement. Here are a few.

An loop, to be executed three times:

```
\repeat \to{3} \do {
   \message{hello there!}
}
```

Looping until the counter reaches some condition, here divisibility by 37:

```
\newcount\tmpcount
\repeat \for{j}
   \until{\tmpcount\j \divide\tmpcount by 37 \noexpand\ifnum\tmpcount=1}
  \do {
    \message{testing \number\j}
}
```

An example of nested loops, where the inner loop uses the loop counter of the outer loop in its bounds:

```
\repeat \for{i} \by{2} \to{10} \do
  {\repeat \for{j} \from{\i} \by{3} \to{18} \do
   {\message{(\number\i.\number\j)}
  }}
```

That's it, folks. I have a hard time imagining that someone could want yet more from a loop macro, but if you can think of something, just let me know.