The outlining macros are now complete. There is one small problem: One might occasionally need to use the pound sign for its normal TEX function of marking a parameter in a \def or \halign, inside an outline. We can make that possible by providing a macro that temporarily changes the category code of the pound sign back to normal.

```
\def\normalpoundsign{%
    \bgroup
    \catcode`\#=6
    \innernormalpoundsign
}%
\def\innernormalpoundsign#1{#1\egroup}%
```

Thus an \halign could be enclosed in \normalpoundsign{...}.

Finally we restore the at sign to its former category code.

```
\catcode`\@=\oldatsigncatcode
```

A Macro Writing Tool: Generating New Definitions

Amy Hendrickson
TEXnology Inc.

Suppose you come upon a situation where you need a macro which will generate another new macro every time it is used. I came upon a solution to this problem and want to share it with TUG readers in case someone would find it an useful macro writing tool, or maybe just find it amusing.

The problem that I ran into that necessitated this kind of macro (it is by no means the only application) had to do with a set of macros that I was writing recently for slide generation: How can you take large chunks of text possibly containing tables, listings, verbatim text, or section headers, and a) print the chunk where it appears in the document, then b) send it to the end of the file to be printed in slide format. (This format would include larger font and baselineskip, possibly be in landscape mode, and have rounded corner edging.)

Since you cannot send a large body of text to an auxiliary file, the solution seemed to be to write one macro which would generate as many definitions as there were chunks of text to be made into slides, and send only the control sequence and slide formatting information to an auxiliary file. The auxiliary file can then be input at the end of the original file, and the definitions that were made earlier in the file will produce the slides.

But how can one generate such a series of definitions, each with a new name? The solution involves using the letters of roman numerals as the name of the each new macro. A counter is advanced to produce a new roman numeral each time the macro is used. With the right macro expansion, the roman numerals will be interpreted as a sequence of letters, and a new sequence of letters will be available each time.

For instance, say we set the counter equal to 637 to start, and advance it by one every time the macro is used. The first set of letters that will become a control sequence will be \dcxxxvii, the second \dcxxxviii, etc.

To make certain that these letters have not already been used in a definition, we can also supply, following the roman numeral, a sequence of letters that does not change, and thus make the possiblity of renaming a previously defined control sequence very small. That is the function of the \unique definition below.

Here is some code, showing how \newdefs can be used to define #1 as a new definition every time the macro is used.

```
\newcount\definitionnum  \definitionnum=2001

\def\newdefs#1{\advance\definitionnum by 1
  \def\unique{\the\definitionnum ZZZZ}
  \expandafter\gdef
    \csname\romannumeral\unique\endcsname{#1}}
```

In use,

```
\newdefs{This is a chunk of text}
```

will produce

```
\gdef\mmiiZZZZ{This is a chunk of text}
```

a control sequence that can be called for later in the file in whatever application it might be useful.