UTILITY MACROS

Patrick Milligan
Lynne A. Price

BNR INC.,
subsidiary of Bell-Northern Research, Ltd.

As a part of our experience with the creation and use of \TeX macros, several small but useful macros have been written to aid in the creation of large macro packages. This article highlights these utility macros, and their usage.

Font Definition

In order to facilitate the definition and use of fonts not declared in \TeX, a macro called \fontdef was written to declare a font and define a macro to invoke it. This macro takes three arguments:

\fontdef {<Font Code>}{<Font Name>}{<Macro Name>}

For example, the standard definition of \rm from \TeX would look like:

\fontdef {a}{cmr10}{\rm}

In addition, we have adopted the convention that our standard macro packages never use uppercase letters as Font Codes, so that users always know these letters can be used for any special fonts declared in a specific document.

The source for \fontdef follows:

\def\fontdef#1#2#3{\font #1=#2 \def #3{\curfont #1}}

Counter Value Comparison

To extend the macros \neg and \ifzero given in Appendix X of the \TeX manual, we have created a macro called \ifieq which tests equality between two values (which can either be constants or counters). \ifieq takes four arguments:

\ifieq {<Value 1>}{<Value 2>}{<Then Clause>}{<Else Clause>}

This macro uses counter 9 as a scratch counter. We have found that always having a scratch counter available is a reasonable way to implement general counter arithmetic. The following example of \ifieq compares counter 0 equal to 1:

\ifieq {\count0}{0}{Is one}{Isn't one}

The source for \ifieq follows:
Utility Macros

The macros \neg and \ifzero are copied from Appendix X.
We have added \ifeq. Unlike other similar macros, \ifeq expects
its arguments to be values, so if a counter is used it must be
specified (e.g., \count3 instead of 3) and constants are permitted.
\def\neg\if\advance\count#1\else\neg\fi
\def\ifzero\if\ifnum#1=0\then\else\fi\fi\setcounter{#1}{0}
\def\ifeq\if\ifnum#1=#2\then\else\fi\fi\setcounter{#1}{#2}
\def\ifpos\if\ifnum#1>0\then\else\fi\fi\setcounter{#1}{#2}
\def\fifneg\if\ifnum#1<0\then\else\fi\fi\setcounter{#1}{#2}

Pseudo Counters

In writing large macro packages which keep track of page, chapter, section, subsection, table, and figure numbers, it is likely that the ten counters provided by \TeX will not be adequate. One alternative is to use macros to hold counter values. By using \xdef and \setcount, it is possible to convert counters into macros, and vice versa. To facilitate the advancing of pseudo counters, a macro called \advcounter was written. This macro takes two arguments:

\def\advcounter{<Pseudo Counter>}{<Advance Value>}

Counter 9 is used as a scratch counter in \advcounter. The following example "sets" counter \pagenum to 1, then "advances" it by 2:

\def\pagenum{1} \advcounter \pagenum {2}

The source for \advcounter follows:

% Macro to advance pseudo-counters (i.e., macros defined to be integers
% in order to bypass \TeX's limited number of counters)
\def\advcounter#1#2{\setcounter{#1}{\value{#1}+#2}\xdef\num{\count{#1}}}

Uppercase Roman Numerals

\TeX's facility for lowercase Roman numerals is useful in a variety of applications. However, it is not obvious how to obtain uppercase Roman numerals! Assuming that counter 0 holds a negative value the intuitive attempt

\uppercase{\count0}

doesn't work since \uppercase see \count0 as a single, unexpanded token, not as a token list consisting of the Roman numeral equivalent. By using \xdef, we can force the expansion of the negative counter to the Roman numeral string, allowing \uppercase to produce the desired uppercase Roman numeral. Thus, the (non-obvious) sequence

\xdef \num{\uppercase{\count0}} \num

is what we want!

We have written two macros to return the upper or lower case Roman equivalent of a positive counter. The macro \roman returns a lowercase Roman numeral, and the macro \Roman returns an uppercase Roman numeral. Both of these macros use counter 9 as a scratch counter.

The source for these macros follows:
Utility Macros

\% Provide for converting positive counters to upper or lower case Roman
\def\roman#1{\setcount#1\-\count\+\count9}
\def\Roman#1{\setcount\-\count\+\count9
-\count\+\def\uppercaseroman{\uppercase{\count9}}\uppercaseroman}