"Hey — it works!"

Jeremy Gibbons
University of Auckland
jeremy@mail.cs.auckland.ac.nz

Welcome once again! I'm a little short of space this issue, so I'll be brief and let the articles speak for themselves. We have two followups to John Langer's article on self-expanding acronyms, and an article by Charles Wells on how to refer back from a bibliography to occurrences of citations in the document.

Please note that my email address has changed.

More on self-expanding acronyms

Paul A. Thompson
Case Western Reserve University
pat@po.CWRU.Edu

In TTN 3,4:10, there is a little description of a self-expanding macro for acronyms. While handy, there are several problems with the method described there: such acronym definitions are difficult to type routinely, as the parentheses nest in an odd manner; they are clunky to read; and a compromise between 'define 1 time' and 'define all times' may be desired—you may wish to expand the acronym three times, or refer to the page on which the acronym has been first noted.

Here is an acronym macro which solves all of these 'problems'. Acronyms are defined as arguments to a macro, `\acromake`. Definitions are easier to read. The macro takes three arguments: the name of the command that will be defined, the acronym, and the expanded, full term (in that order). The user may define the number of full references desired (via the macro `\ACRcnta`), and a number of 'acronymized' references which include the page number of the first fully-stated reference (via the difference between `\ACRcntb` and `\ACRcnta`).

For example, the definition

```
\acromake{Acro}{AA}{Another Acronym} \Acro; \Acro; \Acro; \Acro.
```

yields: Another Acronym (AA); AA (see Page 1); AA; AA.

The macro provides an example of generating macros with macros, a process which sounds more arcane than it actually proves to be. As such, I asked for assistance from `comp.text.tex`. I received it from Donald Arseneau, David Carlisle and Peter Schmitt, to whom I wish to extend a note of appreciation.

For every acronym, the macro generates the acronym macro and a macro which acts like a counter for acronym use. Since counters are quite rare, it is irresponsible to use a counter for every acronym—rather, a macro is generated which fills this function. If the acronym macro is called `\foo`, then the counter macro is called `\fooz`. Thus, I would avoid calling any acronym 'fooz', as this may be confoozing.

```
\newcommand{\ACRcnta}{1}%  full acronym generations
\newcommand{\ACRcntb}{2}%  ACRcntb-ACRcnta times acronym & pageno
\newcounter{util}
\newcommand{\addtomacro}[2]{% % like \addtocounter, but for a macro
  \setcounter{util}{#1}\addtocounter{util}{#2}\edef#1{\theutil}}
\newcommand{\acromake}[3]{%
  \ifnum\ACRcnta<1\renewcommand{\ACRcnta}{1}\fi%
  \expandafter\newcommand\csname#1z\endcsname{0}%
  \expandafter\newcommand\csname#1\endcsname{%
    \ifnum\csname#1z\endcsname=0\label{acro:#1}\fi%
    \expandafter\addtomacro\expandafter{\csname#1z\endcsname}{1}%
    #3 (#2)% actual acronym printed here
    \ifnum\csname#1z\endcsname=\ACRcnta%
      \expandafter\renewcommand\csname#1\endcsname{%
        \expandafter\addtomacro\expandafter{\csname#1z\endcsname}{1}%
        #2%
        \ifnum\csname#1z\endcsname>\ACRcntb\else
          \ (see Page \pageref{acro:#1})%
        \fi%
        }%
    \fi%
    }%
}
```

Evolving acronyms

Peter Schmitt
University of Vienna
SCHMITT@Pap.UniVie.AC.AT

In the last issue (TTN 3,4:10), John V. Langer pointed out, that, when dealing with acronyms, it is convenient to use a macro that redefines itself after the first call. In this issue Paul A. Thompson pursues this idea further and presents a macro that automates the definition of macros for acronyms. Using macros which change their meaning twice he offers the possibility to add (for a certain number of calls) a reference to the explanation of the acronym (given when used for the first time). His code is straightforward. The number of calls (advanced each time the acronym is used) is stored in a macro which is checked and compared with the desired number of repetitions.

I want to describe an alternative approach to the basic tool, i.e., an 'evolving' (self-redefining) macro. Instead of doing arithmetic it uses the most primitive way to store the number of calls—this number is represented by a sequence of 'strokes' (in the form of `\or`-tokens). Then `\ifcase` is used to 'count' them and determine whether the prescribed bound has been arrived. This approach is implemented by `\perform`. A second macro, `\make`, is used to set up the evolving macro.

The macro `\make` takes four arguments: the name of the evolving macro (`#1`), the initial meaning (`#3`), the final meaning (`#4`), and, of course, a number (`#2`) (or a control sequence expanding to a number) which determines how often the initial meaning has to be used. It expands to a call of `\perform`, passing suitable arguments:

```
\def \make #1#2#3#4{%
   \expandafter\def\csname#1\endcsname{\perform{#1}{#2}{}{#3}{#4}}}
```

The macro `\perform` takes five arguments. It first performs `#4` (used to pass the current, i.e., initial meaning of the macro) and then redefines the macro (whose name is `#1`). There are two cases between which `\ifcase` decides: If the number `#2` of desired calls is greater than the number of previous calls (represented by a corresponding number of `\or`-tokens in `#3`) plus 1 (an additional `\or`) then the `\else`-case is executed and defines the macro to perform another call of `\perform`, inserting an additional `\or` into the third argument. If not, i.e., if the initial meaning has already been repeated `#2` times, the last `\or`-case sets the macro to its final meaning `#5`.

```
\def \perform #1#2#3#4#5{%
     #4%
     \ifcase #2 #3\or
             \expandafter\def\csname#1\endcsname{#5}%
       \else \expandafter\def\csname#1\endcsname
                         {\perform{#1}{#2}{\or#3}{#4}{#5}}%
         \fi}
```

The following example illustrates the usage:

```
\make{chameleon}{2}{yes}{no}\chameleon\chameleon\chameleon\chameleon
```

produces 'yesyesnono'.

Macros which change their meaning several times can be obtained either directly by repeated use of `\perform`, or (as observed by Jeremy Gibbons) by recursive use of `\make`. For instance, `\Make` (defined below) defines a macro (`#1`) which performs `#3` for `#2` calls, then `#5` for `#4` calls, and `#6` afterwards.

```
\def \Make #1#2#3#4#5#6{%
   \expandafter\def\csname#1\endcsname
      {\perform{#1}{#2}{}{#3}{\perform{#1}{#4}{}{#5}{#6}}}}
```

or

```
\def \Make #1#2#3#4#5#6{%
   \make{#1}{#2}{#3}{\make{#1}{#4}{#5}{#6}\csname#1\endcsname}}
```

In both cases,

```
\Make{cham}{2}{yes}{3}{maybe}{no}\cham\cham\cham\cham\cham\cham\cham\cham
```

results in 'yesyesmaybemaybemaybenonono'.

Cross references in the bibliography

Charles Wells
Case Western Reserve University
cfw2@po.CWRU.Edu

Some time ago on `comp.text.tex` someone asked for LaTeX code that would print the pages on which each entry in the bibliography was referred to. The following style file, `xbib.sty`, does that.

To use this style, you have to enclose each bibliographical entry in braces. Thus, instead of

```
\bibitem{LG} Leonard Gillman, {\em Writing Mathematics Well}. MAA, 1987.
```

you should write

```
\bibitem{LG}{Leonard Gillman, {\em Writing Mathematics Well}. MAA, 1987.}
```

Note that if you later decide not to use `xbib.sty`, these extra braces will cause no harm.

A message is printed at the beginning of the bibliography explaining the list of pages after the entries. This message is contained in the command `\xbibmessage` and can be changed. To remove it, put `\def\xbibmessage{\relax}` in the preamble *after* reading in `xbib.sty`.

By redefining `\xbibpointer`, you can cause the cross reference list to refer to sections (define it to be `\thesection`) or other subheadings.

For example, citing Gillman's book [1] here (and making no other citations) would produce the bibliography:

> At the end of each entry, the pages on which that entry is cited are listed in parentheses.
>
> [1] L. Gillman, *Writing Mathematics Well*. MAA, 1987. (4)

If you want to use `xbib.sty` with BibTeX you must modify the `.bst` file you use. Most `.bst` files define the two functions `output.bibitem` and `fin.entry` as in the left-hand column below; they should be changed as shown in the right-hand column.

Original BIBTEX	Modified BIBTEX

`FUNCTION {output.bibitem}`
`{ newline$`
`  "\bibitem[" write$`
`  label write$`
`  "]{" write$`
`  cite$ write$`
`  "}" write$`
`  newline$`
`  ""`
` 'output.state before.all :=`
`}`
	`FUNCTION {output.bibitem}`
	`{ newline$`
	`  "\bibitem[" write$`
	`  label write$`
	`  "]{" write$`
	`  cite$ write$`
	`   "}{" write$`
	`  newline$`
	`  ""`
	`  'output.state before.all :=`
	`}`

`FUNCTION {fin.entry}`
`{ add.period$`
`  write$`
`  newline$`
`}`
	`FUNCTION {fin.entry}`
	`{ add.period$ write$`
	`   "}" write$`
	`  newline$`
	`}`

I recommend that you rename the resulting file by prepending an 'x' to its name. Thus `alpha.bst` would become `xalpha.bst`. As far as I know, if you use such modified files no difference will result in the output if you don't use `xbib.sty`.

This style file must be used with LATEX version 2.09 or later. It will not work with earlier versions. It also redefines the LATEX commands `\@lbibitem`, `\@bibitem` and `\@citex`, which are used in the definition of `\bibitem` and `\citex`. Therefore, it may not work if you use a style file that redefines these terms (the article and book styles are ok).

```
% This file is xbib.sty

% The following command causes the page numbers to be listed in
% parentheses, separated by commas, with no period.
\def\showcites#1{\@ifundefined{xb@#1}{\relax}{(\@nameuse{xb@#1})}}

% The following two commands cause the reference list to be printed
% at the end of the bibliographic entry after a 1 em space.
\def\@lbibitem[#1]#2#3{\item[\@biblabel{#1}]
   \if@filesw{\def\protect##1{\string ##1\space}%
      \immediate \write\@auxout{\string\bibcite{#2}{#1}}}\fi
   \ignorespaces{#3}\hskip 1em\showcites{#2}}

\def\@bibitem#1#2{\item
   \if@filesw \immediate\write\@auxout
      {\string\bibcite{#1}{\the\value{\@listctr}}}\fi
   \ignorespaces{#2}\hskip 1em\showcites{#1}}
```

```
\def\addcite#1#2{%
   \@ifundefined{xb@#1}%
      {\global\@namedef{xb@#1}{#2}\global\@namedef{xbb@#1}{#2}}%
      {\def\@tempc{#2}
       \xdef\@tempb{\csname xbb@#1\endcsname}
       \ifnum\@tempb=\@tempc\else
          \global\@namedef{xbb@#1}{#2}
          \xdef\@gtempa{\@nameuse{xb@#1}, #2}
          \expandafter\xdef\csname xb@#1\endcsname{\@gtempa}
       \fi
      }%
}
\def\xbibpointer{\thepage}

\@input{\jobname.xbi}
\if@filesw
   \newwrite\@xbibfile \immediate\openout\@xbibfile=\jobname.xbi
\fi

\def\@citex[#1]#2{%
   \if@filesw\immediate\write\@auxout{\string\citation{#2}}\fi
   \def\@citea{}%
   \@cite{\@for\@citeb:=#2\do
      {\@citea\def\@citea{,\penalty\@m\ }%
      \@ifundefined{b@\@citeb}
         {{\bf ?}\@warning{%
                    Citation `\@citeb' on page \thepage\space undefined}}%
      \if@filesw{%
         \let\thepage\relax \def\protect{\noexpand\noexpand\noexpand}%
         \xdef\@gtempa{%
            \write\@xbibfile{\string\addcite{\@citeb}{\xbibpointer}}}}%
         \@gtempa\fi
      \csname b@\@citeb\endcsname}}%
   {#1}}

\def\thebibliography#1{%
   \section*{References\@mkboth{REFERENCES}{REFERENCES}}
   \xbibmessage
   \list{[\arabic{enumi}]}%
      {\settowidth\labelwidth{[#1]}\leftmargin\labelwidth
       \advance\leftmargin\labelsep\usecounter{enumi}}
   \def\newblock{\relax}
   \clubpenalty4000\widowpenalty4000
   \sfcode`\.=1000\relax}
\def\xbibmessage{At the end of each entry, the pages on which
   that entry is cited are listed in parentheses.}
```