“Hey — it works!”

Jeremy Gibbons
Department of Computer Science
University of Auckland, New Zealand
`jeremy@cs.aukuni.ac.nz`

Greetings. I have just taken over the column *Hey — it works!* from Christina Thiele. This column is designed for little tricks — even ones that seem trivial or inelegant — that others may find useful. If you have any such items, send them in to me at the address above.

The first piece in this issue, from Navindra Gambhir, continues from Claudio Beccari's note in the previous issue on aligning decimal points; it is an improvement in that you don't need to manually add padding after the decimal place, nor (as with `decalign.sty`) do you need to specify the number of decimal places used.

My colleague Hans Guesgen's piece shows how to insert LaTeX material into a bibliography in a disciplined way.

The third piece highlights the fact that the column title could be taken in two ways. The intended meaning is one of practicality (“Hey, don't knock it — it works!”), but it could also be taken as one of surprise (“Well, bless my cotton socks, it *works*!”). Alan Jeffrey's note on pattern matching falls squarely into the latter category.

But enough talk — on with the show!

Lining up decimal points

Navindra Gambhir
Cornell University, Ithaca NY
`gambhir@cs.cornell.edu`

A method to line up the decimal points in a column of a numeric table that uses active `?`'s to produce blank spaces is explained on p. 241 of *The TeXbook*. The question whether there is a “nicer” way to do this was recently raised in the newsgroup `Info-TeX`. Hence, for example, one would like to be able to typeset the table

Jan. 8	0.125	2.25	.0625
Feb. 13	$128.5 \pm .1$		$<.01$
Mar. 23	16	256	.5
Apr. 29	.25	.0	32

by simply typing

```
$$\dpalign{\indent#\hfil&&\qquad\span\dpaligntemplate\cr
    Jan.~8&   0.125&       2.25&  .0625\cr
    Feb.~13&  128.5\pm.1&  &      <.01\cr
    Mar.~23&  16&          256&   .5\cr
    Apr.~29&  .25&         .0&    32\cr}$$
```

The trick, which is based on an idea also suggested on p. 241 of *The TEXbook*, is to make the "`.`" behave locally like an alignment tab within `\dpaligntemplate`. To handle numbers that do not have a decimal point, `&` is also made to behave locally like *two* alignment tabs with an empty column in between. Some care is needed to get the timing right when changing the category codes.

```
\def\dpalign#1#{% #1 is <box specification>, if any
  % change the catcodes before reading <alignment material>
  \dpalignglobaldefs \dodpalign{#1}}
\def\dodpalign#1#2{% #1 is <box specification>
                   % #2 is <alignment material>
  \halign#1{#2\crcr}}
\def\dpaligntemplate{\dpaligndefs%
                     \hfil$##$\tabskip=0pt&$.##$\hfil}
{\catcode`\.=\active \catcode`\&=\active \catcode`\|=4
  % | is temporary alignment tab
  \gdef\dpalignglobaldefs{\catcode`\.=\active \chardef.=`\.%
    \catcode`\&=\active \let&=|}% active . and & act normally
  \gdef\dpaligndefs{\def.{|}\def&{|\omit\hfil|}}}
```

The macro `\dpalign` should be used within a group, e.g., inside a `\vbox`, or within `$$`'s, as in the above example. Such grouping can also be made a part of the macro by adding it to the definitions of `\dpalign` and `\dodpalign` in the obvious places. Also note that `\dpaligntemplate` sets `\tabskip` to `0pt`, so it has to be reset explicitly, if necessary, after `\dpaligntemplate` in the preamble.

Headings in reference lists

Hans W. Guesgen
University of Auckland
hans@cs.aukuni.ac.nz

Since I want to use my BIBTEX bibliographies also for reference lists in non-scientific papers like annual reports, resumes, etc., I am interested in incorporating headings and commenting text into a reference list. Of course, it is always possible to edit the resulting `.bbl` file to get the job done, but this is annoying if you want to produce such papers on a regular basis. So I looked for another way of adding interspersed headings to the reference list.

The trick I came up with might look awful, but it certainly works. I created an additional `.bib` file that contains all the headings I want to use in the reference list. The headings are defined as usual LATEX commands within `@misc` BIBTEX entries:

```
% file headings.bib
@misc{books, note = "\bibheading{Monographs}%"}
@misc{inbooks, note = "\bibheading{Papers in Books}%"}
@misc{injournals, note = "\bibheading{Journal Articles}%"}
```

It is essential to have a `%` as the last character in the `note` field to comment out the punctuation character produced by BIBTEX.

After defining the set of potential headings, I wrote a LATEX style file that specifies what to do with them. Typesetting the reference list with labels wasn't appropriate any more, since I didn't want the headings ending up with labels. Thus, I redefined the `\thebibliography` command, replacing the enumerating list environment with a simple one, and I defined how the headings are to be typeset in the new environment:

```
% bibheadings.sty, for bibliographies with interspersed
% headings; use simple, rather than enumerated, list.
\def\thebibliography#1{%
   \list{}{\leftmargin0em \itemsep0.5\baselineskip}%
   \def\newblock{\hskip .11em plus .33em minus .07em}%
   \clubpenalty4000\widowpenalty4000 \sfcode`\.=1000\sloppy}
% typeset the headings in the same format as subsections
\def\bibheading#1{\subsection*{#1}}
```

Now I can cite (via `\nocite`) the entries of my actual bibliography and the pseudo-bibliography containing the headings in the order they are supposed to appear in the bibliography. Whenever a pseudo-entry is cited, for example, by `\nocite{books}`, the contents of its `note` field are included in the `.bbl` file:

```
\begin{thebibliography}{1}
...
\bibitem{books}
\bibheading{Monographs}%.
...
\end{thebibliography}
```

The following is an example of LATEX source for producing a reference list with headings.

```
\documentstyle[bibheadings]{article}
\begin{document}
\section*{Relevant Publications}
\bibliography{headings,mybib}
\bibliographystyle{unsrt}
```

```
\nocite{books}                                 % heading
\nocite{guesgen:89:consatbook}                 % book
\nocite{guesgen:hertzberg:92:dynconsbook}      % book
\nocite{injournals}                            % heading
\nocite{guesgen:ea:92:jpdc}                    % article
\end{document}
```

Processing the source with LaTeX and BibTeX results in the following:

Relevant Publications

Monographs

H.W. Guesgen. *CONSAT: A System for Constraint Satisfaction.* Research Notes in Artificial Intelligence. Morgan Kaufmann, San Mateo, California, 1989.

H.W. Guesgen and J. Hertzberg. *A Perspective of Constraint-Based Reasoning.* Lecture Notes in Artificial Intelligence 597. Springer, Berlin, Germany, 1992.

Journal Articles

H.W. Guesgen, K. Ho, and P.N. Hilfinger. A tagging method for parallel constraint satisfaction. *Journal of Parallel and Distributed Computing*, 16:72–75, 1992.

`\howto` pattern match in TEX

Alan Jeffrey
University of Sussex
alanje@cogs.sussex.ac.uk

TEX's pattern matching ability is quite powerful, but is also quite infuriating. It allows you to pattern-match against an arbitrary string, but when it fails, it can do nothing but produce an error message. For example, if you say:

```
\def\foo(#1,#2){\message{(#1 and #2)}}
\foo(a,b) \foo(c) \par
```

Then you will get a message `(a and b)` but you will then get an error message telling you about a `Runaway argument`. It would be nice if you could use TEX's pattern-matching ability, but provide more than one pattern, and have TEX use the latest definition that matches. This note describes such a command, so you can say:

```
\howto\oneof(#2){#2}
\howto\oneof(){}
\howto\oneof(#2,#3){#2 \vee \oneof(#3)}
$$ \oneof(a,b,c,d) $$
```

and get the result:

$$a \vee b \vee c \vee d$$

This definition says that any argument matching the pattern (x, y) should produce $x \vee$ `\oneof`(y), any argument matching the pattern () should produce nothing, and any argument matching the pattern (x) should produce x.

This is all done using TeX's pattern-matching ability (which explains the rather odd fact that the parameters have to be numbered starting at `#2`, since a pattern may fail by having something before parameter `#2`, which has to be matched by parameter `#1`). If you say:

`\howto\foo(`*pattern*`){`*expansion*`}`

then we look to see if `\foo` is already defined, and if it is not, we define it to produce an error message. Then we increment the counter `\actioncount` to get n and define the command `\action-`n so that `\doaction\action-`$n(x)$ expands out to "*expansion*" if x matches *pat*, and to the previous definition of `\foo`(x) otherwise.

A wonderfully incomprehensible piece of programming, but hey — it works!

```
\def\xcs#1#2{\expandafter#1\csname#2\endcsname}
\newcount\actioncount
\def\howto#1(#2)#3{%
   \xcs\ifx{\expandafter\gobble\string#1}\relax
   \def#1{\erroraction#1}\fi
   \advance\actioncount by 1\relax
   \expandafter\defaction\expandafter{#1}{#3}{#2}%
   \def\temp##1##2##3##4##5##6##7##8##9{\def\temp{(#2)}}%
   \temp\empty\empty\empty\empty\empty\empty\empty\empty\empty
   \edef#1{\noexpand\doaction\xcs\noexpand
      {action-\the\actioncount}\temp}}
\def\defaction#1#2#3{%
   \xcs\def{action-\the\actioncount}##1(#3){\applyto{#1}{#2}}}
\def\gobble#1{}
\def\applyto#1#2#3{#3{#1}{#2}}
\def\doaction#1(#2)(#3){#1(#3)\success(#2)\failure(#3)}
\def\success#1#2(#3)\failure(#4){#2}
\def\failure#1#2{#1}
\def\erroraction#1(#2){%
   \errmessage{I don't know how to \string#1(#2)}}
```