"Hey — it works!"

Jeremy Gibbons
University of Auckland
jeremy@cs.aukuni.ac.nz

Welcome to *"Hey — it works!"*. 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, please send them in to me at the address above.

Paul Hafner, in the Department of Mathematics and Statistics here at Auckland, pointed out a problem with Hans Guesgen's macros in the last issue for interspersing headings in bibliographies. Hans was working from an older version of LaTeX, and an update to LaTeX in January 1992 broke Hans' macro; the fix is to add the definition

```
\def\@bibitem#1{\item\if@filesw \immediate\write\@auxout
       {\string\bibcite{#1}{}}\fi\ignorespaces}
```

to `bibheadings.sty`.

The first piece in this issue is a brief note from Klaus Lagally about Peter Schmitt's margin mark macros in TTN 1,3. Next we have an item by Claudio Beccari on fine adjustments to the spacing of alignments, expanding on his own comment in TTN 2,1. We conclude with a short piece from Charles Wells on giving arbitrary labels to equations.

More on margin marks

Klaus Lagally
University of Stuttgart
lagally@informatik.uni-stuttgart.de

Here are a few comments on "Marking lines in the margin", TTN 1,3 p. 12 and TTN 1,4 p. 10, by Peter Schmitt:

The macros as given will inhibit hyphenation. This can be corrected by redefining:

```
\long\def\readword#1 {#1\strut\putmark\space\next\relax}
```

If the margin mark is a `\vrule` of sufficient height and depth the `\strut` is no more necessary, not even at the beginning of a paragraph, as TeX will now be in horizontal mode anyway.

How to redefine `\putmark` to get the mark into the right margin is left as an exercise to the interested reader.

Correct spacing for tables and arrays

Claudio Beccari
Politecnico di Torino
beccari@polito.it

Every LaTeX user knows how to insert vertical spacing in tables, arrays and equation arrays by means of the optional argument of the `\\` command. I suppose that most users do not like to explore the inner depths of LaTeX, so that it is possible that some users do not realize that the `\\` command behaves quite differently in equation arrays compared with tables and arrays.

In the `eqnarray` environment (with or without asterisk) the `\\` command with an optional argument — a dimension given in square brackets — actually inserts that vertical space in addition to the default row spacing defined by the format file (in fact this default spacing is given by the `\jot` dimension, which is defined in `lplain.tex`, not in `book`, `report`, `article`, or `letter`, nor in the size-related subsidiary files); of course, anybody can assign this `\jot` a different value, but most users simply don't do it.

The `\\` command optional argument behaves in a different way in the `tabular` and `array` environments; this optional argument is used to increment the depth of the strut that implicitly is inserted in the first cell of every row and in the only cell produced by means of the `\multicolumn` command. Therefore it is impossible or difficult to control vertical spacing after every `\hline` and `\cline` command, and after those rows that contain at least one cell whose depth is greater than the implicit strut depth.

With the set of LaTeX or plain TeX predefined commands, in the first case there is no workaround, except, perhaps, inserting a whole row of empty cells, which may result in too much white space. In the second case it is possible to proceed empirically until one finds the right amount of extra space to insert as the optional argument to the `\\` command. In any case if you try to modify the implicit strut dimensions by redefining `\arraystretch`, you end up with a table or an array that contains too much white space.

Both cases, on the other hand, may be dealt with very nicely if some new struts are defined and inserted in the right places (in addition to the implicit strut) in order to control the vertical space very finely. In my tables and arrays I generally use the following two struts, but in special cases I might define several similar invisible objects:

```
\newcommand\T{\rule{0pt}{2.6ex}}                % Top strut
\newcommand\B{\rule[-1.2ex]{0pt}{0pt}}          % Bottom strut
```

The 'top strut' ensures that its row is at least 2.6ex high; the 'bottom strut' ensures that it is at least 1.2ex deep.

Compare the following two ruled tables; you will notice in the second one that horizontal rules are not too close to capital letters and that fractions (even if they are set in textstyle and therefore are smaller than usual) do not require guesswork in order to keep them at a reasonable distance from the following rule.

Laplace transforms		
$f(t)$	(p)	no struts
$\delta(t)$	1	optional arg.
$\cos\omega_0 t$	$\frac{p}{p^2+\omega_0^2}$	

Laplace transforms		
$f(t)$	(p)	with struts
$\delta(t)$	1	optional arg.
$\cos\omega_0 t$	$\frac{p}{p^2+\omega_0^2}$	

Notice that a bottom strut in the second table is inserted in the denominator of the fraction, not to the side of the fraction, otherwise its extra space would be masked by the fraction depth.

The second table is obtained with the following commands:

```
\newcommand\T{\rule{0pt}{2.6ex}}              % Top strut
\newcommand\B{\rule[-1.2ex]{0pt}{0pt}}        % Bottom strut
\begin{tabular}[t]{|c|c|l|}
   \hline
   \multicolumn{3}{|c|}{Laplace transforms\T\B}\\\hline
   $f(t)$       \T & $(p)$      \B & with struts  \\\hline
   $\delta(t)$     & $1$        \T & optional arg.\\[.5ex]
   $\cos\omega_0t$ & $\frac{p}{p^2+\omega_0^2\B}$ &    \\\hline
\end{tabular}
```

The `\\` command optional argument is used after the third row, because it does not contain oversized cells, otherwise it would be better to control vertical spacing with struts only.

Instead of locally defining the top and bottom struts, as done in the example, it is possible to put the definitions in the preamble, or, even better, to define a command that defines them, so that the definition may be kept local to a particular environment or group:

```
\newcommand\setTBstruts{\def\T{\rule{0pt}{2.6ex}}%
                        \def\B{\rule[-1.2ex]{0pt}{0pt}}}
```

so that it is possible to get them to work with a simple call to `\setTBstruts` only where it is needed.

Labelling equations by names

Charles Wells
Case Western Reserve University
cfw2@po.cwru.edu

Sometimes you want to give one or two of the main results of your paper a name. If the result is in the form of an equation, you might want it to be labeled by the name you gave it instead of by a number. The following four lines define two new commands, `\labeq` and `\elabeq`, that produce a labeled equation for which you choose the label. You can also refer to it later by its label or its page number. The equation does not use up an equation number: if the equation before the labeled equation is number n, the equation after the labeled equation will be number $n+1$.

```
\makeatletter
\def\labeq#1{\def\@currentlabel{#1}\label{#1}$$}
\def\elabeq{\eqno{\mbox{(\@currentlabel})}$$}
\makeatother
```

You would use it like this:

```
\labeq{MAIN}
x^2+y^2=z^2
\elabeq
```

which would print as

$$x^2 + y^2 = z^2 \qquad \text{(MAIN)}$$

You could refer to it by writing

```
See equation~(\ref{MAIN}) on page~\pageref{MAIN}.
```

This would print as "See equation (MAIN) on page 4." Note that you must put the parentheses in yourself.

This definition is a genuine hack since it depends on internal LATEX code that is not guaranteed to be the same in future revisions.