Hints and Tricks

'Hey — It Works!'

Jeremy Gibbons

[Editor's note: Welcome once again to *"Hey — it works!"*, a column devoted to (LA)TEX tips and tricks. This was intended for the last (alas, never published) issue of *TTN*, and Jeremy should not be held directly responsible for it at this point in time. We hope to include this column regularly in *TUGboat*.]

— — * — —

Jonathan Fine of Cambridge, UK wrote in response to Allan Reese's article in *TTN* 4,2. Allan explained why inserting `\lowercase` into an `\equal` test, as in

```
\renewcommand{\NL}[1]
  {\def\thisletter{#1}%
   \ifthenelse{\equal{%
    \lowercase{\thisletter}}%
        {\lowercase{\thatletter}}}%
     {\\}%
     {\\[\smallskipamount]%
      \global\def\thatletter{#1}}%
   #1}%
```

did not produce a case-insensitive string comparison — the `\lowercase` happens too soon, even with judicious uses of `\expandafter`.

Jonathan observed that the problem can be solved by applying the `\lowercase` around the *definitions*, rather than the uses, of `\thisletter` and `\thatletter`:

```
\renewcommand{\NL}[1]
  {\lowercase{\def\thisletter{#1}}%
   \ifthenelse{\equal{\thisletter}%
    {\thatletter}}%
     {\\}%
     {\\[\smallskipamount]%
      \lowercase{\global\def\thatletter{#1}}}%
   #1}%
```

We also have three new items this issue. The first, is about typesetting long division problems; Barbara Beeton submitted the idea and the crucial part of its solution to me a couple of years ago, and I've rewritten it in LATEX. I've also shown Donald Arseneau's amazing macros for doing the whole long division problem automatically, given just the dividend and divisor. The second article, by Christine Thiele, shows how to produce two small separately-numbered figures side by side; I used it in my column in *TTN* 4,1. The final article, by Dennis Kletzing, is a followup to his presentation at the TUG Annual Meeting in Florida in July 1995; he shows here how to typeset automatically many short items of differing sizes in a grid layout. Enjoy!

— — * — —

Long division

Barbara Beeton
American Mathematical Society
`bnb@ams.org`

Donald Arseneau
Tri-University Meson Facility
`asnd@erich.triumf.ca`

Barbara Beeton was asked by a secretary for help with typesetting a long division problem, such as

$$\begin{array}{r@{}l}
 & 949 \\ \cline{2-2}
13\big) & 12345 \\
 & 117 \\ \cline{2-2}
 & 64 \\
 & 52 \\
 & 125 \\
 & 117 \\
 & 8
\end{array}$$

Barbara solved the secretary's problem, and was particularly proud of the use of the parenthesis. I've taken her idea and rephrased it in terms of LATEX's `tabular` environment (I'm afraid Barbara is a die-hard `plain` TEXie!), which makes placing the digits relatively straightforward.

The long division above was produced by

```
\newdimen\digitwidth
    \settowidth\digitwidth{0}
\def~{\hspace{\digitwidth}}
\def\divrule#1#2{%
  \noalign{\moveright#1\digitwidth%
      \vbox{\hrule width#2\digitwidth}}}
13\,\begin{tabular}[b]{@{}r@{}}
      949 \\ \hline
      \big)\begin{tabular}[t]{@{}l@{}}
        12345 \\
        117  \\ \divrule{0}{4}
        ~~64 \\
        ~~52 \\ \divrule{2}{3}
        ~~125 \\
        ~~117 \\ \divrule{2}{3}
        ~~~~8
   \end{tabular}
\end{tabular}
```

The macro `\divrule#1#2` produces a rule the same width as `#2` digits, indented by the width of `#1` digits. (In most fonts, all digits have the same width.) Notice the `\big)` between the divisor and the dividend, and also the use of `~` as an active character, producing a space the size of a digit; this should all be done within a group, in order that the redefinition of `~` is not global. Notice also the use of nested tabulars, one bottom-aligned and one top-aligned, for placing the various parts correctly.

Of course, TeX is quite capable of doing long division itself. Donald Arseneau has posted the following macros on comp.text.tex:

```
\newcount\gpten % power-of-ten, tells which
                % digit we're doing
\newcount\rtot
     % running total -- remainder so far
\newcount\scratch

\def\longdiv#1#2{%
     % long division: #1/#2;  integers only
 \vtop{\offinterlineskip
   \setbox\strutbox\hbox{%
   \vrule height 2.1ex depth .5ex width0ex}%
   \def\showdig{$\underline{%
    \the\scratch\strut}$\cr\the\rtot\strut\cr
      \noalign{\kern-.2ex}}%
   \global\rtot=#1\relax
   \count0=\rtot\divide\count0by#2%
        \edef\quotient{\the\count0}%
%
   % make list macro out of digits in quotient:
   \def\temp##1{\ifx##1\temp\else
     \noexpand\dodig ##1%
        \expandafter\temp\fi}%
   \edef\routine{\expandafter%
        \temp\quotient\temp}%
%
   % process list to give power-of-ten:
   \def\dodig##1{%
      \global\multiply\gpten by10\relax}%
   \global\gpten=1\relax\routine
    % to display effect of one digit
    % in quotient (zero ignored):
   \def\dodig##1{%
      \global\divide\gpten by10\relax
      \scratch =\gpten
      \multiply\scratch  by##1\relax
      \multiply\scratch  by#2\relax
      \global\advance\rtot-\scratch \relax
      \ifnum\scratch>0 \showdig \fi
      % must hide \cr in a macro to skip it
   }%
   \tabskip=0pt
   \halign{\hfil##\cr % \halign for entire
                      % division problem
     $\quotient$\strut\cr
     #2$\,\overline{\vphantom{\big)}}%
     \smash{\raise3.5\fontdimen8%
        \textfont3\hbox{$\big)$}}%
     \mkern2mu \the\rtot}$%
         \cr\noalign{\kern-.2ex}
     \routine \cr
        % do each digit in quotient
}}}
```

Given these macros, the long division

$$\begin{array}{r} 949 \\ 13\big)\overline{12345} \\ \underline{11700} \\ 645 \\ \underline{520} \\ 125 \\ \underline{117} \\ 8 \end{array}$$

is produced simply by typing
    \mbox{\longdiv{12345}{13}}.

— — ∗ — —

Side-by-side figures

Christina Thiele
Ottawa, Canada
cthiele@ccs.carleton.ca

Something I recently had to do in LaTeX was provide two figures side-by-side. There are actually two types of situation: one is where the figures are related, and numbered 1a and 1b, for example; the other is where they are numbered separately, as 1 and 2. The first instance, sub-figures, can be handled by getting subfigure.sty, by Steven Douglas Cochran, from the nearest CTAN site. The second instance can be done with regular LaTeX commands. I have to thank Barbara Beeton who forwarded mail from comp.text.tex from January 1994 when the subject arose there; also thanks to the exchanges posted by Andrew Justin Caird, Gabriel Zachmann and Tim Murphy (who pointed the way to subfigure.sty). Gabriel has recent informed me that another, more elegant, solution would be to use David Carlisle's tabularx.dtx, obtainable from CTAN in tex-archive/macros/latex/packages/tools/.

The bare-bones template I extracted from that correspondence was as follows (the references were to photos of Cree leggings, in an article on same):

```
\begin{figure}
  \begin{minipage}{5cm}
    \vspace{7cm}
    \caption{Woman's leggings}
  \end{minipage}
\hfill
  \begin{minipage}{5cm}
    \vspace{7cm}
    \caption{Man's leggings}
  \end{minipage}
\end{figure}
```

I sent this notion off as a possible item for *TTN*. But before I had a chance to get this simple version written up, a new application was found, in Charles Wells' piece on "Cross references in the bibliography" (*TTN* 4,1:7). There, the code to produce the comparison between original and modified BibTeX put the minipages inside a tabular, making for an easy way to do two-column work without doing line-by-line two-columning. The minipage environment might be worth having an extended tutorial on its uses; I suspect there's a lot can be done with it.

— — * — —

Enumerated arrays

Dennis Kletzing
Stetson University, Florida
kletzing@bliss.stetson.edu

Typesetting the solutions manual for a book frequently involves creating an enumerated list involving many short answers. These answers are usually enumerated across a row. The simple way to do this is to start typing, with the result that the enumeration counter is not aligned vertically from one row to the next. Moreover, the ends of lines frequently wrap to the next line and do not look good. Surely there must be a better way to do this!

I started thinking abut this after I saw a message posted on comp.text.tex from someone asking what the secret was for writing macros. The person responding mentioned several things and concluded with the words "think boxes". If we "think boxes", it is not difficult to put together a macro that typesets enumerated arrays.

Basically, each item will be typeset in an \hbox which consists of three boxes; one for the label, one for the label separation, and one for the item itself. We first choose a basic unit length \mitemwidth to measure the length of items, a label width, and a label separation width. Extend the \mitemwidth to \mtotalwidth, which includes the label width and label separation. Finally, create a box, \mitembox, to hold the item to be typeset.

```
\newdimen\mitemwidth
    \mitemwidth=.1\textwidth
\newdimen\mitemlabelwidth
    \mitemlabelwidth=2em
\newdimen\mitemlabelsep
    \mitemlabelsep=0.5em
\newcount\mitemtempcount
\newcounter{mitemcounter}
\newdimen\mtotalwidth
     \mtotalwidth=\mitemwidth
\advance\mtotalwidth
     by\mitemlabelwidth
\advance\mtotalwidth
     by\mitemlabelsep
\newbox\mitembox
```

To ensure that each entry is typeset consistently, we use this rule: measure the width of the entry, including label and separation. Then typeset it in a box whose width is the next largest multiple of the unit length. Thus, if an entry requires 2.3 inches and the unit length is 1 inch, 3 unit lengths will be required. Here is the macro:

```
\def\mitem#1{%
      \refstepcounter{mitemcounter}%
  \setbox\mitembox=\hbox{%
    \hbox to \mitemlabelwidth{%
      \hfil\arabic{mitemcounter}.}%
    \hskip\mitemlabelsep\hbox{#1}%
       \hskip0pt}%
  \mitemtempcount=\wd\mitembox%
    \advance\mitemtempcount
        by\mtotalwidth%
    \advance\mitemtempcount by -1%
    \divide\mitemtempcount
        by \mtotalwidth%
  \setbox\mitembox=\hbox
        to\mitemtempcount\mtotalwidth{%
    \box\mitembox\hfill}%
  \leavevmode\box\mitembox\hskip0pt}
```

[Notice the idiom $(x + y - 1) \div y$ for rounding-up integer division of x by y, in terms of rounding-down division $\div$; also, the dimension \mtotalwidth is being used as a \count. –jg]

For example:

1. $x = 3$	2. $y = -2$	
3. $x = 7$	4. 8	
5. circle	6. $y = 2x^2$; parabola	
7. $2x^2 + y^2 = 9$; ellipse		8. circle
9. $x^2 + y^2 = 9$		10. $x + y = 3$
11. line	12. $x = 21$	13. $y = 5$
14. $z = -5$	15. $x = 7$	16. $y = -9$
17. 2	18. 5	19. 6
20. -1	21. 9	

This was set using the code

```
\begin{flushleft}
\mitem{$x=3$}
\mitem{$y=-2$}
\mitem{$x=7$}
\mitem{8}
\mitem{circle}
\mitem{$y=2x^2$; parabola}
\mitem{$2x^2+y^2=9$; ellipse}
\mitem{circle}
...
\end{flushleft}
```

In this example the \mitemwidth is set to 0.1\textwidth. The user should experiment with different values of \mitemwidth to see how the shape of the array changes. This macro has the advantage that if changes are made to the entries, all items are renumbered and arranged appropriately. One disadvantage is that if an entry takes more than a single line, it will not wrap the line. Also, there are some situations when we wish to wrap several lines within the entry. For these cases, there is a more elaborate package called `multienum.sty` which sets each item in a `\parbox` of width 1, 1/2, 1/3, or 1/4 of the `\textwidth`, wrapping the entry if necessary. Interested readers may contact me for a copy of it. Finally, I want to say a special word of thanks to Jeremy Gibbons for his help in putting together this version of the multienumerate macro.

◇ Jeremy Gibbons
School of Computing and
Mathematical Sciences
Oxford Brookes University
Gipsy Lane, Headington
Oxford, OX3 0BP UK
`jgibbons@brookes.ac.uk`