"Hey — it works!"

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

Welcome once again to *"Hey — it works!"*. This column is devoted to tips and tricks for (LA)TEX and friends—little things that might help readers solve typesetting problems of their own.

Last issue, we had two follow-ups to John Langer's note on self-expanding acronyms in TTN 3,4. Having actually read the last issue of TTN, I see that there is an announcement (p. 22–23) of Matt Swift's `Frankenstein` LATEX2ε package, whose `tmacros.sty` style file defines *abbreviation macros*, solving the same problem.

In this issue, we have another nice article from the prolific pen of Donald Arseneau, this time on automatically sizing a middle delimiter *à la* `\left` and `\right`. We also have an article by Alvar Wenzel on labelling counters that are not normally 'labelable', and one by Allan Reese on automatically adding extra space between rows of a long table.

Please note that my email address has changed *back* again.

Automatic sizing of left, right, and middle

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

Question: Is it possible to automatically resize a group of *three* delimiters the way `\left` and `\right` work for two?

Answer: In general, no; at least not easily. But if the middle delimiter is a vertical line it is possible to use the `\vrule` command instead of the vertical bar character. A vertical rule automatically expands to the height and depth of its surroundings, as in

`\left[ {\sum} \vrule {\int} \right]` $\Longrightarrow \quad \left[{\sum} \middle| {\int} \right].$

This is a bit cramped, so it is best to insert some space around the `\vrule` (using `\,` for example). It is even possible to get fancy with macros and have the '`|`' character interpreted as '`\vrule`', as in this macro for Dirac's bra-ket notation:

```
{\catcode`\|=\active
   \gdef\Braket#1{\left<\mathcode`\|"8000\let|\bravert {#1}\right>}}
\def\bravert{\egroup\,\vrule\,\bgroup}
```

so that

$$\texttt{\textbackslash Braket\{\textbackslash phi|\textbackslash frac\{\textbackslash partial\}\{\textbackslash partial t\}|\textbackslash psi\}} \quad \Longrightarrow \quad \left\langle \phi \middle| \frac{\partial}{\partial t} \middle| \psi \right\rangle .$$

The size of the vertical line may not be exactly the same as the left and right delimiters, but it is never far off.

Some fine points worth mentioning: (1) The `\mathcode` assignment makes the `|` character 'active' locally within the `\left \right`, but lets it behave normally outside. (2) The `\let` is also local to allow different use of an active `|` elsewhere, perhaps in tables. (3) `\left<` is just like `\left\langle` (*The TEXbook*, p. 156), but executes faster.

If the middle delimiter is not a vertical line, then middle-resizing is still possible, but you are stuck with using multiple `\left` and `\right` pairs and a few `\phantom`s.

Labelling footnotes in LATEX

Alvar Wenzel
Universität Heidelberg, Germany
wenzel@awi-nov.awi.uni-heidelberg.de

If a footnote[1] contains important information it can be nice to reference it. But with LATEX this cannot be done by writing `\footnote{\label{whatever}...}`, because that way `\label` will catch only the pagenumber correctly, while the `footnote` counter itself is not referencable.

Searching for a solution I first found a rather complicated one, but inspired by Charles Wells' contribution in TTN 2,3 about labelling equations by names, and thanks to helpful comments by the editor of this column now there is a much better solution: A new command `\footnotelabel` is defined by

```
\makeatletter
\def\footnotelabel{%
   \edef\@currentlabel{\p@footnote\thefootnote}%
   \label}
\makeatother
```

and used like in `\footnote{\footnotelabel{something}...}`. It is referenced by `\ref{something}` and `\pageref{something}`. Indeed the footnote on page 2 has number 1 (and these numbers were generated by LATEX).

[1] Like this short one.

This method can be generalized for self–defined counters (e.g., for a new counter `proposition` used to enumerate a new environment which is also called `proposition`). Just define an analogue to `\thefootnote` for example by

```
\def\theproposition{\thesection--\arabic{proposition}}
```

and then replace every occurrence of '`footnote`' in the above definition of the macro `\footnotelabel` by '`proposition`' to get a new labelling command named `\propositionlabel`.

Dividing a table alphabetically

R. Allan Reese
University of Hull
r.a.reese@ucc.hull.ac.uk

To make a long table easier to read, it helps to insert some extra vertical space every few lines. The following defines a macro `\NL` to be called at the end of every line, which uses a counter to insert space every N lines; it uses the macro `\ifthenelse` from the standard `ifthen` style.

```
\newcounter{lineNo} \setcounter{lineNo}{2}
\newcommand{\NL}
   {\addtocounter{lineNo}{-1}%
    \ifthenelse{\value{lineNo}=0}%
       {\\\noalign{\smallskip}\setcounter{lineNo}{2}}%
       {\\}}%
```

For example, the table

Keyword	**Description**
ability	obsolete *DOS* integrated package
babel	multi-language support in TEX
BASIC	generic name for line-based command languages
zterm	modem utility for Macs
1-2-3	spreadsheet. V2 and V3 still in use

was produced by the code

```
\begin{center}\begin{tabular}{ll}
   \bf Keyword & \bf Description\\
      \hline \noalign{\smallskip}
   ability & obsolete {\it DOS} integrated package\NL
   babel   & multi-language support in \TeX\NL
   BASIC   & generic name for line-based command languages\NL
   zterm   & modem utility for Macs\NL
```

```
   1-2-3   & spreadsheet. V2 and V3 still in use\\
      \hline
\end{tabular}\end{center}
```

(Of course, if the table is long enough that extra spacing is needed, you would probably use something like `longtable.sty`.)

As soon as I showed this to a colleague he asked for space not after N entries but at points where the initial letter of the keyword changed. This was a challenge! I had not used the `ifthen` option before but assumed it would easily allow comparison of letters. In the event I was surprised how difficult it was to compare one letter (token) with another and faced an additional problem of making the test case-insensitive. Finally there was a 'keyword' that did not start with a letter.

Making the conditional space depend on the initial character of the next line requires the comparison of two tokens. There is a test in `ifthen.sty`—`\equal{string1}{string2}`—so putting the 'current' letter into a macro string and comparing it with the parameter of `\NL` was straightforward.

At this point I found that the test was case-sensitive. TeX has the command `\lowercase`; *The TeXbook* describes this as converting the text using character codes *and then rereading that text*. Inserting `\lowercase` commands into the string test however did not give the desired result. Experiments with `\expandafter` still did not convert the strings before comparing them. The reason is in the small print: the arguments of the `\equal` test are expanded fully as macros, but `\lowercase` is an operator that works on tokens. In Knuth's analogy, strings are expanded in TeX's gullet but the `\lowercase` operation would not be applied until its stomach.

The solution was to make the case conversion into a macro expansion. The bad news is that this requires defining macros equivalent to every character to be converted. The good news is that this makes the process very general and independent of the language and character set. There was also, by chance, one line in the table starting with a non-letter; this demonstrates how to define the transformation for any character.

The test shown here reads the first non-space character after `\NL`, turns this into a TeX command word and compares this with the previous value. Whenever the letter changes some vertical space is inserted in the table and the new 'current letter' is saved. Finally, the parameter is replaced in the input stream.

```
% Macros to compare letters regardless of case.
\def\A{a} \def\B{b} ... \def\Z{z}
\def\a{a} \def\b{b} ... \def\z{z}
\expandafter\def\csname 1\endcsname{1} % Special case for 1-2-3 !

\def\currentletter{\a}
\newcommand{\NL}[1]
```

```
{\ifthenelse{\equal{\csname#1\endcsname}{\currentletter}}%
    {\\}%
    {\\[\smallskipamount]%
     \global\def\currentletter{\csname #1\endcsname}}%
 #1}%
```

With this revised definition of `\NL`, the same code as before produces the table

Keyword	**Description**
ability	obsolete *DOS* integrated package
babel	multi-language support in TEX
BASIC	generic name for line-based command languages
zterm	modem utility for Macs
1-2-3	spreadsheet. V2 and V3 still in use

with extra space every time the first ‘letter’ changes.