Hints and Tricks

'Hey — It Works!'

Jeremy Gibbons

Welcome to *Hey — it Works!*. This column is devoted to interesting, elegant or just surprising tips and tricks for (LA)TEX. The title has two interpretations: surprise — 'well, bless my cotton socks, it actually works!' — and pragmatism — 'don't knock it, it does the trick'. Articles fitting either interpretation are welcome, whether arcane wizardry or simple but useful techniques; the overriding criterion is brevity and elegance.

This column ran in *TEX and TUG News* until the demise of that newsletter in 1995. Barbara Beeton has kindly agreed to continue it in *TUGboat*, which has absorbed the newsletter. During the intervening period, I have moved from New Zealand to the UK. I have also collected all back-issues of the column since I took it over from Christina Thiele in 1993, and made them available through the URL

`http://www.brookes.ac.uk/~p0071749/hiw/`

In this issue we have three contributions. Donald Arseneau, as ever, has a nice piece showing how to remove a counter from the list of counters to be reset at the start of each section-like unit in LATEX; this is more difficult than adding a counter, but Donald presents a very elegant solution. Ramón Casares shows how to change the default thickness for `\hrule`s and `\vrule`s from the standard hard-wired `0.4pt`. The final article is by yours truly, and explains how to define a 'small verbatim' environment, as used in this column.

◇ Jeremy Gibbons
School of Computing and Mathematical Sciences
Oxford Brookes University
Gipsy Lane, Headington
Oxford, OX3 0BP, UK
jgibbons@brookes.ac.uk
http://www.brookes.ac.uk/~p0071749/

1 Removing a counter from a reset list

By default, the LATEX `report` and `book` classes reset equation and other numbers at the start of each chapter, but once I needed to number equations (figures, etc.) sequentially throughout a report. I could have created an entire document class (copying from `report.cls`) but with the counters defined differently, reducing `\newcounter{equation}[chapter]` to just `\newcounter{equation}`, and likewise for other counters. That's a bit ridiculous, though, when it is the only change I want to make! What is more convenient is to *remove* the counter resets.

LATEX keeps the list of counters that are to be reset with each section-like ⟨*unit*⟩ (where ⟨*unit*⟩ is "chapter," "section," etc.) in the macro `\cl@`⟨*unit*⟩. Clearly, what I needed was to remove the equation counter from the list `\cl@chapter`. There is a simple (internal) LATEX macro called `\@addtoreset` to add a counter to this list, but there is none for removal; so I wrote one of the form

```
\@removefromreset{equation}{chapter}
```

Johannes Braams wrote on this topic in *TUGboat* Vol. 15 (Dec. 1994, p. 496) and explains the functioning of the reset list, but his solution is more complex than necessary, with nested looping, whereas the following definition efficiently redefines the list in a single scan.

```
\def\@removefromreset#1#2{%
 % preserve \@elt (probably unnecessary)
  \let\@tempb\@elt
 % put what we want to remove in \@tempa:
  \expandafter\let\expandafter\@tempa
    \csname c@#1\endcsname
 % define \@elt to check for removal:
  \def\@elt##1{\expandafter\ifx
    \csname c@##1\endcsname\@tempa\else
 % else, reinsert without execution:
    \noexpand\@elt{##1}\fi}%
 % redefine list as itself with removal:
  \expandafter\protected@edef
    \csname cl@#2\endcsname
    {\csname cl@#2\endcsname}%
 % restore \@elt
  \let\@elt\@tempb}
```

How does this work? The list `\cl@`⟨*unit*⟩ is a sequence of commands of the form `\@elt{`⟨*ctr*⟩`}`, one for each counter ⟨*ctr*⟩ (equation, e.g.) to be reset at the start of each ⟨*unit*⟩ (chapter). The command `\@removefromreset{equation}{chapter}` temporarily defines `\@elt{equation}` to disappear (i.e., to expand to nothing) but `\@elt{section}` to remain unchanged (i.e., to expand to itself). Then `\cl@chapter` is simply defined as itself, with these definitions in effect!

When numbering in a continuous sequence, I don't like the chapter number as a prefix to the equation numbers, so I redefine the equation numbering with

```
\def\theequation{\arabic{equation}}
\@removefromreset{equation}{chapter}
```

along with removals for the figure and table counters, if desired. These lines and the definition of `\@removefromreset` should be put in a style file. The definition of `\@removefromreset` is on CTAN in the file `macros/latex/contrib/other/fragments/removefr.tex`, which you should copy into any local `.sty` or `.cls` file where you would like to use `\@removefromreset`.

◇ Donald Arseneau
TRIUMF
4004 Wesbrook Mall,
Vancouver B.C.
Canada V6T 2A3
asnd@reg.triumf.ca

2 Default Rule Thickness

Sometimes the default rule thickness, `0.4pt`, is not the one you need. The solution is to write, for example, `\vrule width 1pt` instead of `\vrule`. But this does not work if the `\vrule`s are hidden inside macros you do not want to modify. When I was in such a situation my first reaction was to look for a ⟨*dimen parameter*⟩ in *The TeXbook*, p. 274. As usual in these cases, I saw some parameters I had never imagined, but nothing resembling the `\rulethickness` I needed[1].

Fortunately in TeX (almost) everything can be done.

```
\let\oldhrule=\hrule
\let\oldvrule=\vrule
\def\rulethickness{\afterassignment
 \dorulethickness\dimen0 }
\def\dorulethickness{\edef\hrule
  {\oldhrule height\dimen0 }%
 \edef\vrule
  {\oldvrule width\dimen0 }}
```

From now on a declaration as `\rulethickness=1pt` makes the default thickness of all rules equal to `1pt`. The next line is an example of a `1pt \hrule`.

Note that now, if you want a `0.4pt` rule you have to write `\vrule width 0.4pt` as expected. Note also that you could omit the '`=`' in the assignment; `\rulethickness 1pt` or `\rulethickness1pt` are also valid.

◇ Ramón Casares
Telefónica de España
rcg@tid.es

[1] There is a `\fontdimen` in maths fonts controlling default rule thickness, but that applies only to rules in maths mode. –jg

3 Small verbatim material

In order to keep verbatim material (such as the code for macros in this column, or example programs on OHP slides) to a reasonable length, it is often desirable to set it in `\small` size rather than the normal size. What you *can't* do to achieve this is to define a `smallverbatim` environment by

```
\newenvironment{smallverbatim}
  {\small\verbatim}{\endverbatim}
```

— an approach that would work if the task were instead to define, say, a 'small quotation environment'. The `verbatim` environment is a strange beast, quite unlike other environments; it is not ended by 'executing' `\end{...}` as other environments are, but rather by finding *exactly* the 14 characters '`\end{verbatim}`' in the file. Finding a macro that expands to these characters is not enough. (The perils of macro expansion languages!)

One apparent solution is to forego the specialized environment, and do it manually each time:

```
Here is the previous paragraph.
\begin{small}\begin{verbatim}
Verbatim material.
\end{verbatim}\end{small}
Here is the next paragraph.
```

Unfortunately, this doesn't work well. By the time the `\begin{verbatim}` has ended the previous paragraph, the size has already been set to `\small` and the `\baselineskip` reduced accordingly; the result is that the entire previous paragraph gets set with too little leading. Here is an example, using `\scriptsize` verbatim and a paragraph without ascenders or descenders to emphasize the effect:

Here is tHe Previous ParaGraPH. Here is tHe Previous ParaGraPH. Here is tHe Previous ParaGraPH.

```
Here is some
verbatim material.
```

Here is tHe next ParaGraPH. Here is tHe next ParaGraPH. Here is tHe next ParaGraPH.

If you look carefully, you will find many published papers and books exhibiting this problem.

You could avoid this scrunching up by leaving a blank line (or a `\par`) before the `\begin{small}`, thereby ending the previous paragraph before the size change. However, it is all too easy to forget that blank line, making the document rather fragile (a blank line before a `verbatim` environment appears to be ignored under normal circumstances). A better solution than this is needed.

For the earlier editions of this column, which appeared in *TeX and TUG News* and used the old

LaTeX 2.09, I had to resort to writing my own environment, mimicking the standard `verbatim` environment but changing to `small` size between ending the previous paragraph and starting the verbatim material. However, LaTeX 2_ε provides a very convenient hook for just such a change: the macro `\verbatim@font`. The default definition is

```
\def\verbatim@font{\normalfont\ttfamily}
```

but you can make all your verbatim material small by replacing the `\normalfont` by `\small`. This, however, has the unfortunate side effect of making `\verb` material also appear small, which may not be what you want. A final solution can be obtained by redefining the `verbatim` environment so that it changes `\verbatim@font` just for that single instance of the environment:

```
\let\VERBATIM\verbatim
\def\verbatim{%
  \def\verbatim@font{\small\ttfamily}%
  \VERBATIM}
```

which is what I have done for this column.

◇ Jeremy Gibbons
Oxford Brookes University
jgibbons@brookes.ac.uk