Hints and Tricks

'Hey — it works!'

Jeremy Gibbons

Welcome to *'Hey — it works!'*, a column devoted to (LA)TEX and META tips, tricks and techniques. Any short and elegant TEX-related items are warmly received.

In this issue, as usual, we have three articles. One is my own, and provides a macro for margin notes that you can switch on and off. The second is by Andreas Scherer, and shows how to draw smooth graphs using METAPOST's `graph` package. The third is by Ramón Casares, and demonstrates how to disable TEX's rule for deciding whether a '`.`' ends a sentence.

Last issue (Vol. 19, No. 4) we had an article by Christina Thiele showing how to produce ornamental rules out of ordinary symbols, using `\cleaders`. The final paragraph demonstrated how to alternate two symbols, but to get an odd number of symbols properly laid out involved switching to `\leaders` instead of `\cleaders`, with the result that the rule is no longer centred within the requested width:

'×÷×÷×÷×÷×   '

Barbara Beeton responded to point out that it is not hard to recentre the rule: first trim its width to the actual printed width, and then 'manually' centre this trimmed rule within the requested width:

```
\def\bordertwo#1#2#3{{%
  \setbox1=\hbox{#1}%
  \setbox2=\hbox{#1#2}%
  \dimen0=#3\advance\dimen0 by -\wd1
  \divide\dimen0 by \wd2
  \multiply\dimen0 by \wd2
  \leavevmode
  \hbox to #3{\hfil#1\hbox to \dimen0
    {\leaders\hbox{#2#1}\hfil}\hfil}%
}}
```

This generates

' ×÷×÷×÷×÷× '

instead.

◇ Jeremy Gibbons
CMS, Oxford Brookes University
Gipsy Lane, Headington
Oxford OX3 0BP, UK
jgibbons@brookes.ac.uk
http://www.brookes.ac.uk/~p0071749/

1 Switchable marginal notes

I often find it convenient, when working on the draft of an article, to annotate it in the margin with reminders of facts to check, corrections to make and so on. Of course, this is what LATEX's `\marginpar` macro is for. However, I would also like to be able to switch off the annotations, for example when I am distributing the draft article to an audience for whom the annotations are inappropriate or irrelevant. I don't want to have to edit the document to remove the annotations one by one; that's just too much trouble. To solve this problem I wrote a simple macro for 'switchable marginal notes'.

The following definitions should be put into a style file, say `margnote.sty`:

```
\newif\ifmarginnotes \marginnotestrue

\def\marginnotestyle
  {\scriptsize\itshape\raggedright}

\def\marginnote#1{%
  \@bsphack
  \ifmarginnotes
    \marginpar{\marginnotestyle#1}%
  \fi
  \@esphack}
```

Then marginal notes can be used by including

```
\usepackage{margnote}
```

in the document preamble.

The first line defines the marginal note switch. Marginal notes are turned on by default, but they can be turned off simply by saying

```
\marginnotesfalse
```

after the `\usepackage` declaration. (They can even be turned off and on mid-document.)

The second section specifies the style of marginal notes; by default, they are in a smaller size, italic, and set ragged right, but this can be changed by using `\renewcommand`.

The remainder of the file defines the marginal note macro itself. This takes a single argument, the text of the note, and sets it using `\marginpar` if marginal notes are turned on. For example,

```
\marginnote{This is a marginal note.}
```

produces the note in the margin here. The macros `\@bsphack` and `\@esphack` are internal to LATEX; they ensure that an entity like a marginal note or label definition does not introduce any extra space into a paragraph, independently of whether or not it is attached to a word.

This is a marginal note.

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

2 Smoothing augmented paths in METAPOST

The user manual of the METAPOST `graph` package states that neighbouring points of a path created with the `augment` macro are connected by straight line segments. Depending on the application, it may be more suitable to draw a smooth curve through the set of points on the path, using the '`..`' operator. This can be achieved easily.

Let the input be an external data file `hiw.data` containing several pairs of coordinates, each pair on a separate line:

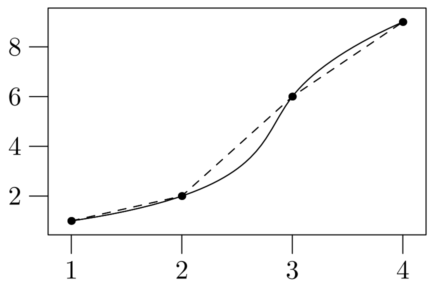
```
1 1
2 2
3 6
4 9
```

The following METAPOST code creates a (jagged) path by calling `augment` as the third argument to the `gdata` routine:

```
input graph;

beginfig( 1 );
  draw begingraph( 5cm, 3cm );
    path p;

    gdata( "hiw.data", c,
        augment.p( c1, c2 ); );

    gdraw p dashed evenly;
    gdraw (point 0 of p
      for i = 1 upto length p:
        .. point i of p
      endfor);

    pickup pencircle scaled 3pt;
    for i = 0 upto length p:
      gdraw point i of p;
    endfor;
    pickup defaultpen;

  endgraph;
endfig;
end.
```

This path is `gdrawn` the first time as a dashed line, depicting the default behaviour of `augment`. The "Hey, it works!" effect is achieved in the next four lines by `gdrawing` a (temporary) smooth version of the same path. This is done directly as an argument to `gdraw`; no new variables are needed. Note how this is done in a simple `for` loop running over the points of the paths, applying the '`..`' operator in between.

Together with the control points displayed as heavy dots, the result of this code is shown in the following picture.



METAPOST automatically scales the x- and y-axes, adds a frame (whose size was set in the `begingraph` command), and attaches tick marks and labels.

METAPOST's `graph` package will not generate cyclic paths, but nevertheless a similar approach can be used to draw a smooth version of a cyclic polygon:

```
draw (for i = 0 upto (length p - 1):
    point i of p ..
  endfor cycle);
```

◇ Andreas Scherer
Rochusstraße 22–24
52062 Aachen, Germany
andreas.scherer@pobox.com

3 Every point a period

The rule used by TeX to decide whether a point is a period ending a sentence (so it will stretch the following space) or is just indicating an abbreviation is, for a simple mind like mine, too complicated. And it fails more frequently than expected when my text is full of ugly acronyms. So I have devised an alternative scheme.

Basically the idea is that every point is a period ending a sentence, so when I want to use a point in any other circumstance I have to protect the space that follows it, if any. If I want this space to be breakable then the solution is to write a backslash between the point and the space, that is '`.\␣`'. If, on the other hand, I want this space to be unbreakable then the solution is to write a tilde between the point and the space, that is '`.~␣`'. Easy, isn't it?

The code to achieve this is as follows:

```
\count255='A
\loop
  \sfcode\count255=1000
  \ifnum\count255<'Z\advance\count255 1
\repeat
\def~{\nobreak\ \ignorespaces}
```

Note that I have appended a `\ignorespaces` to the tie mark definition (so in fact a space after a tilde is ignored).

◇ Ramón Casares
Telefónica de España
r.casares@computer.org