# Q & A

## JUST PLAIN Q&A: The Russians are Coming!

Alan Hoenig

We've received three good questions, all of them from Andrei Khodulev of Mir Publishers in Moscow (Soviet Union). I hope more people will respond. Contact the TUG office or me directly (`ajhjj@cunyvm` on Bitnet, or by telephone, (516) 385-0736) with your TeX questions.

### Cutouts Spanning Paragraphs

Mr. Khodulev needs to know how to use `\parshape` to cross several paragraphs. Suppose, for example, he needs to leave space for a 3 cm high cutout. Suppose the paragraph at which the cutout starts is only 24 pt high. Can we ask TeX to somehow continue the effects of `\parshape` across paragraph boundaries? Furthermore, we need to specify the cutout in absolute measure, not using the numbers of lines, as `\parshape` expects. Is this possible?

The answer to this question formed the subject of an earlier article by Tom Reid in *TUGboat* 8, no. 3, 315–320 (November 1987). This article contains many useful TeX hack tricks and deserves to be better known than it is. Tom presents a set of macros which do what the Russians require. We note that his macros don't use numbers of lines. Simply set a box to the required height and width:

```
\setbox\figbox=
 \vbox to 4 cm {\hbox to2cm{\hss}\vss}
```

for example. Tom's macros do the rest.

An alternative solution to the same problem appears in [1].

By the way, TeX can easily be asked to convert from measure to lines. We can determine the lines by "dividing" the given measure by the value of `\baselineskip`. Here's one way to define a macro `\tolines` to do that. In practice, the command `\tolines 5pt` or `\tolines 3cm` places the number of lines equivalent to the measure in the register `\n`. Note that we don't need any grouping symbols in the argument.

```
\newcount\n
\def\tolines{%
 \afterassignment\dotolines
 \dimen0= }
\def\dotolines{\n=\dimen0
 \ifdim\dimen0<0pt \n=-\n \fi
 \converttolines
 \ifdim\dimen0<0pt \n=-\n \fi }
\def\converttolines{%
 \advance\n by\baselineskip
 \advance\n by-1
 \divide\n by\baselineskip }
```

(A brief explanation. The register `\n` stores the results of the calculation. The `\afterassignment` trickery is necessary so we may use the macro as `\tolines 1in` or `\tolines 4.5\baselineskip` without using braces around the argument to the macro. We add one scaled point less than an extra value of `\baselineskip` to make sure we always round up after the divide operation; normally, TeX's `\divide` operation truncates the fractional portion.) Positive values of `\n` refer to lines down the page, while negative values refer to lines up the page.

### Nested Loops

Mr. Khodulev both poses and answers his own question. The problem: something seems wrong if you use nested loops in what would appear to be an obvious way, such as the following.

```
\newcount\iii \newcount\jjj
\iii=0
\loop
\jjj=0
\loop
\message{(\the\iii,\the\jjj) }
\advance\jjj by 1
\ifnum\jjj<3\repeat
\advance\iii by 1
\ifnum\iii<3\repeat
```

The naïve user would expect nine pairs of numbers, but you see only these: (0,0), (0,1), (0,2), (1,3), and (2,4). The difficulty vanishes when one surrounds the inner loop with braces:

```
\newcount\iii \newcount\jjj
\iii=0
\loop
{\jjj=0
 \loop
 \message{(\the\iii,\the\jjj) }
 \advance\jjj by 1
 \ifnum\jjj<3\repeat}
\advance\iii by 1
\ifnum\iii<3\repeat
```

(The output is nine pairs of numbers as you expect.) The `\loop` macro of `plain` TeX is not primitive, and its definition demands that inner loops be grouped as shown.

The moral: Always surround nested loops with curly braces.

### Fontdimens and Physical Fonts

TEX associates a series of parameters with each font, and looks for these values in the `tfm` file. These font dimensions are accessible to a TEX user via the `\fontdimen` command. (Their significance is summarized in tables on pages 433 and 447 of *The TEXbook.*) Mr. Khodulev has uncovered some puzzling behavior when he tries to alter the `\fontdimens` for his own uses.

For the sake of concreteness, we will use `\fontdimen2`, which specifies the normal interword space for a font. Suppose you wanted to increase the interword space for a certain font in special places in the document. You might try (as he apparently did) something like the following.

```
\font\rm=cmr10
\font\specrm=cmr10
\fontdimen2\specrm=9.99pt
```

You might expect that when you typeset using `\rm`, you get the normal interword spacing (3.33 pt), while you would extra large spaces only when using `\specrm`. In fact, after the `\fontdimen` declaration above, any `\fontname` tied to the physical font `cmr10` has its `\fontdimen` changed. As if to add insult to injury, you cannot attempt to surround changes to `\fontdimen` within a group, since `\fontdimen` assignments are *always* global.

The following lines of code present one way of resolving the problem. The font definitions are encumbered with longer names than usual, but the actual of mechanics of changing fonts are relegated to macros with names that closely resemble normal font calls. These macros have been designed to be used so the user thinks they are font calls, and the rare appearance of `\aftergroup` helps make this syntax possible.

```
%% First, fonts.
\font\roman=cmr10
\font\specroman=cmr10
%% Next, the special registers
\newdimen\savedvalue
 \savedvalue=\fontdimen2\roman
\newdimen\specialvalue
 \specialvalue=9.99pt
%% Finally, definitions.
\def\rm{%
 \fontdimen2\roman=\savedvalue }
\def\specrm{%
 \aftergroup\restoredimen
 \fontdimen2\specroman=\specialvalue
 \specroman }
\def\restoredimen{%
 \fontdimen2\roman=\savedvalue }
```

Mr. Khodulev did not specify his need in any more detail, so these macros should be revised as necessary. With these macros and definitions in force, the source text

```
\rm Here is some text.
{\specrm Here is some spaced out text.}
Here is more text, hopefully
back to normal.

\rm Here is more text.
\specrm Here is some spaced out text.
\rm Text is back to normal.
```

produces

Here is some text. Here is some spaced out text. Here is more text, hopefully back to normal.

Here is more text. Here is some spaced out text. Text is back to normal.

### Bibliography

[1] Hoenig, Alan, "Line-Oriented Layout with TEX," in *TEX: Applications, Uses, Methods*, ed. Malcolm Clark. London: Ellis Horwood (1990).

◇ Alan Hoenig
17 Bay Ave.
Huntington, NY 11743 USA
(516) 385-0736
`ajhjj@cunyvm.bitnet`

# Tutorials

## The `\if`, `\ifx` and `\ifcat` Comparisons

David Salomon

Large, small, long, short, high, low, wide, narrow, light, dark, bright, gloomy, and everything of the kind which philosophers term accidental, because they may or may not be present in things,—all these are such as to be known only by comparison.

— Leon Battista Alberti