were changed to to allow greater flexibility in both specifying and reporting the full file names.

Changes were worked out first on an 'archaic' TFX version of METAFONT but have since been transferred to a more recent (TFM) version in a rather straightforward manner. Some modifications in approach have been suggested by work done at Stanford, although the development has been independent.

A METAFONT description is device-independent as well as point-size-independent—within limits: that limit is METAFONT itself with its highly device-dependent routines and its inability to deal (linearly) with the full range of point sizes of interest to the typesetting community. The METAFONT description may be device independent but METAFONT is not, and that sort of generality is not possible without the use of translators to modify METAFONT's output. The consequence of this last is crucial with respect to the creation of a font library, and its implications are discussed elsewhere in this issue.

The larger figure on the previous page is a Cyrillic O produced by proofmode. Vertical unitlines and significant horizontal lines (representing h-height, x-height, axis-height, baseline, and descender depth, from top to bottom) form a grid on which the character is drawn. Points mentioned in the drawing routine are labelled.

In addition to proofmode (which takes work to implement), METAFONT contains a magnification option which is very easy to use. In ordinary font production for the Varian, one runs METAFONT and sets the variable `pixels` to 3.6. This value represents the number of pixels per point on that machine (increased by 30% since we use the Varian for proofing). To obtain the other character shown, one simply fools the Varian into thinking that one point (the printer's unit) contains 3.6*15 pixels. For the Stanford METAFONT this is done by specifying `mode=-1` (to tell METAFONT that you want a Varian font and that a magnification factor is coming) and `mag=15`. The same sort of procedure would work for any other output device for which METAFONT can produce characters.

Although significant points are neither indicated nor labelled with this magnification option, the figure obtained is of value in design. Subtleties of strokes are more likely to stand out here than in our Varian output of proofmode. It seems to us, for example, that the vertex at the bottom of the letter is more apparent in the smaller than in the larger figure.

* * * * * * * * * * * *

Warnings & Limitations

* * * * * * * * * * * *

Uppercase Update; Fickle Fonts

Last issue's warning about `\uppercase` has now been rendered obsolete by a change in TEX, whereby dimensions (`.5em`) and function words (`for`, `after`) are recognized in any combination of upper- and lower-case letters (`.5EM`, `For`, `aFtEr`). This change was made in Don Knuth's own version of TEX on February 27, 1981, and is described in the errata list among the "Extensions since last printing". But check the status of your local version before changing your macros!

Welcome though this change may be, it lays another trap: A macro for formatting entries in an index included the specification "... `\hangindent 10pt #1` ..." and an index entry "Forecasting" was rendered as "casting", indented just a bit more than the specified hanging indent. Don explains it thus:

"The word '`for`' is allowed after '`\hangindent`'; e.g.,

```
\hangindent 3pt for 5
```

and moreover you can use letters as constants as in

```
\char e     (=\char'145)
```

Then `\hangindent 3pt Forecasting` means `\hangindent 3pt for '145 casting`

Likewise, the word '`plus`' or '`minus`' will be gobbled after `\hskip 1pt ...`"

To disarm the trap, add a couple more braces to the offending macro: "... `\hangindent 10pt{}#1 ...`"

. . . . . .

Probably most sites use a version of TEX that has preloaded fonts, and probably most sites preload the fonts specified in `basic.tex` (see the TEX manual, Appendix B, for details). But some users, for whatever reason, decide to associate different fonts with the one-character names assumed by `basic`. When a file using non-`basic` font names is run through a version of TEX that has preloaded `basic`, the job may run to completion with no warning (the SAIL implementation of TEX gives no warning, but some Pascal versions may—at least the VAX/VMS version does so). TEX will use the metrics of the preloaded fonts, but the spooler will use the fonts requested by the user, and a ragged right margin may be only symptom.

There are two solutions: (1) Go back into the input files and change all conflicting font designators—this can get very messy; or (2) use a version of TEX in which no fonts have been preloaded; such a version, commonly known as "VIRGINTEX", will start up much more slowly than a preloaded version, owing to the greater number of font metric files that must be loaded at run time. The following convention has been adopted at many installations: Preloaded fonts use no capital letters. Thus you are always safe if you introduce a new font called A, B, ..., Z. (Actually, the AMS requires an extended set of fonts, including a full complement of cyrillic fonts in 6 sizes; these are called A, ..., F, but G through Z remain open for special use.)

Barbara Beeton

* * * * * * * * * * * *

MACRO COLUMN

Send Submissions to:
Lynne A. Price
TUG Macro Coordinator
Calma R&D
212 Gibraltar Dr.
Sunnyvale, CA 94086

The macro column is a new regular feature of TUGboat. It is a forum where TEX users can exchange formatting problems (with or without solutions), questions about writing macros, comments on macros published in earlier issues of TUGboat, etc.

* * *

Discussion of macros at the TEX Implementors' Workshop in May included some simple suggestions for increasing portability of macros across TEX sites. First, the excellent suggestion was made that ASCII sites attempt to standardize the characters chosen to replace SAIL delimiters. The $\mathcal{AMS}$-TEX conventions are recommended: ampersand (&) for the tab character, underscore (_) for the subscript indicator, and caret (^) for the superscript delimiter. Second, macro packages typically include several font declarations. Incompatible assignment of font codes makes it difficult for users to select an assortment of macros from different packages. If font codes assigned in a macro file do not correspond to the fonts preloaded by some versions of TEX, strange results can be difficult to explain. There is no total solution to this problem, but it can be minimized. Macro packages should come with documentation describing the fonts and font codes used. When sending files to another installation, users should remember that preloaded fonts differ from site to site. A helpful convention in assigning font codes is to reserve uppercase letters for user declarations and to let standard macro packages use other characters. Patrick Milligan's DefineFont macro described below can be used to automatically assign available font codes.

* * * * * * * * * * * *

Macros on Microfiche

Editor's note: In an effort to hold down expenses, some of the more extensive macro packages in future issues of TUGboat will be published on microfiche, with a summary or introduction to each package included in this column. Authors of macro packages who submit their work for publication here are requested to supply such an introduction along with the camera copy of the package. Because fiche is not as easy to use as paper, an attempt will be made to arrange for the collection and distribution of these macro packages in machine-readable form (probably on magnetic tape); details will be published as soon as they are known. Fiche will conform to the following specifications: negative image (white characters on black), 105mm × 148mm, 24-to-1 reduction ratio, containing 98 frames per fiche.

* * * * * * * * * * * *

ERRATUM: NOFILL PROGRAM

Patrick Milligan
BNR Inc.

There was one subtle error in the program listing of both the SAIL and Pascal versions of the NOFILL program that appeared in TUGboat Vol 2, No. 1. In both programs, the definitions of macros \' and \` were reversed (see pages 90 and 96). As printed, the definitions are correct, but the program source was incorrect. Since the program source was run through NOFILL for publication, the incorrect definitions became correct, but all other uses of ´ (acute accent) and ` (grave accent) were incorrect.

Also, there was some confusion about the table of contents entry on page 136 entitled *NOFILL Program with Pascal Source.* When the two programs were submitted to TUGboat, it was not clear if the SAIL version would be printed, or the Pascal version, or both. The introduction to the SAIL version was appropriate to both versions, but no introduction was prepared for the Pascal code.