Prolegomena toward a font selection scheme

Victor Eijkhout

Department of Computer Science University of Tennessee
104 Ayres Hall
Knoxville, Tennessee 37996, USA

eijkhout@cs.utk.edu

1 Introduction

Most users of plain TEX do not get very sophisticated in their use of fonts. Often they resort to declaring all used fonts explicitly with `\font`. There are some obvious disadvantages to that: it is not possible to switch a whole document in a simple way to a different typeface, or to a different size. As a result, I've seen such phenomena as an article with an abstract in 8 or 9 point, but where the formulas were still in 10 point, or pages of 'magnified' type where the lines were cramped, because the `\baselineskip` was not increased with the type size.

The need for a good font selection scheme is thus quite obvious, but the implementation of one is not.

2 Concepts

2.1 Font parameters

One point that most people agree on is that fonts are characterized by a number of parameters, and that it should be possible in an easy way to select the font that arises from changing just one of those parameters. For instance, switch to italics for emphasis, switch to a smaller size for a footnote, switch to the Greek alphabet to give an erudite – and untranslated, of course – quote from Homer.

However, the exact number of parameters is a point of contention. Appendix E of TEX Users Group, [2], gives macros that are based on size and style (`\rm`, `\it`, et cetera). An obvious third parameter – which Knuth, working exclusively in Computer Modern did not need – is the typeface. In [3, 4] Frank Mittelbach and Rainer Schöpf use an extra parameter, splitting style into 'shape' and 'series'. The series parameter combines the typographical qualities weight and width.

Furthermore, Karl Berry in [1] splits the typeface parameter into 'foundry' and 'typeface family'[1], and Yannis Haralambous suggested that the parameter 'alphabet' is also needed.

An international standard for font properties even exists: in section 8.6 of [5] 23 font properties are given. However, not all of these can be independently varied. For instance, the design copyright and the list of excluded characters are merely informational. The parameters suitable for inclusion in a font selection scheme are here: typeface name, posture (upright, oblique, italic, ...), weight (light, medium, bold, ...), proportionate width (condensed, medium, expanded, ...), structure (solid, outline, ...), and design size. Notably missing is the alphabet parameter.

2.2 More parameters, less parameters; do we have an argument?

To a certain degree, the above font classifications clash less than they seem to. Most parameters are orthogonal: if parameter 1 has p_1 values and parameter 2 has p_2 values, then all $p_1 \times p_2$ values make sense. But in that case we can introduce a new parameter 3 which has that many values. Thus the above schemes differ mostly in how far they split up the variety of fonts, not along what lines.

The number of parameters is to some extent a matter of taste, but a few common sense observations can be made.

- It is possible to live completely without any parameters by using only the control sequences that have been defined with `\font` declarations. This is very inconvenient.
- Certain DTP packages make it easy for you to (produce horrible typography: you simply) switch to 'outline' or 'shadow' (or both) for whatever font you are currently using. This is no reason to introduce independent parameters 'outline' and 'shadow' for every conceivable mutilation of typefaces.
- Some parameters are not likely to get varied: it is hard to conceive how the same font will be used from two different foundries in the same text, other than in texts that write specifically about typeface comparisons.
- Readers may have wondered why the distinction between serif and sans serif has not come up yet. The reason is that this is not an orthogonal category. Most typefaces exist only in seriffed or serifless

[1] It should be noted that this is not a proposal for of a font selection scheme, but for systematic filenames. The number of user parameters can either be more or less than the properties encoded in the file names.

design, but not both. For the occasional exception (Computer Modern, for instance) it is easiest to pretend that the seriffed and serifless subfamilies are simply two typeface families.

3 Implementation

The previous section described some of the issues involved in providing the user with suitable parameters for handling fonts. We must now look at the other side of the question: the parameters have to be translated into the name of a `tfm` file.

Any implementation of a font selection scheme has to pay attention to a couple of points.

- Certain combinations of parameters may not correspond to an existing font. A substitution may have to be performed, and in any case the user has to be told.
- For large sizes there must be a mechanism that makes a reasonable guess as to the file name.
- The memory usage of the font selection scheme should be as low as possible.

The last point can be elaborated a bit: memory usage should increase in a controlled manner if more parameters or more typefaces are added. Adding parameters may seem a bit strange to do, but it makes sense for such parameters as 'structure': suppose all fonts so far have been solid, and suddenly a printer is bought that can make an outline from any given font. Is the amount of memory for the font selection scheme then increased by a small quantity, or is it doubled?

Specifying for every combination of all parameters what file name results (as is done in the font selection scheme in [3]; in order "to implement the four dimensional grid of fonts" it keeps "for every combination of font family/series/shape" a list of size/external name pairs) is in one sense the optimal solution: any system (or absence thereof) of file names can be accommodated. However, the space needed is proportional to the product of all possible values of all parameters.

On the other hand, if file names are systematic (as in the naming scheme in [1], and as in the Computer Modern typefaces and all Bitstream and PostScript typefaces as far as I've seen), then considerable savings are possible: it becomes possible to translate a few parameters jointly into a fragment of the file name, and concatenate such fragments into the full file name. This reduces storage to a higher power root of the original amount[2].

This approach of componentwise translation offers an additional advantage for the 'size' parameter: for PostScript fonts the size can be given procedurally, for instance specifying

```
size: at #1pt
```

thus taking a fixed amount of storage for an infinite number of values.

An entirely modular translation is probably not possible. For instance, the translation of shape and series into a file name fragment is most likely dependent on the typeface, although it may be the same for all typefaces from a certain foundry.

Another impediment to modular translation is the handling of special cases. For instance, for certain typefaces certain styles may not be available, or styles may not be available in all sizes (probably the smaller sizes). However, such exceptions can most likely still be treated in a procedural manner.

References

[1] Karl Berry. Filenames for fonts. *TUGboat*, 11:517–520, 1991.

[2] Donald E. Knuth. *The TEXbook*. Addison Wesley, 1984. reprinted with corrections 1989.

[3] Frank Mittelbach and Rainer Schöpf. A new font selection scheme for TEX macro packages. *TUGboat*, 10:222–238, 1989.

[4] Frank Mittelbach and Rainer Schöpf. The new font family selection – user interface to standard LATEX. *TUGboat*, 11:297–305, 1990.

[5] ISO standard. Information technology – Font information interchange – Part 1: Architecture; international standard ISO/IEC 9541-1, 1991.

[2] For instance, translating three parameters independently makes storage proportional to the sum of the values of the three instead of the product. Thus storage is proportional to the cube root of what it would be in a fully explicit scheme.