Production notes

Karl Berry

The most TEXnically unusual part of this article, and of the entire issue, was handling the rare characters shown in the footnote on the first page and the rundown of circle-slash characters on the next two pages. Although they could have been inserted as small images, the author (Chuck Bigelow) sent me fonts including them, so I wanted to try typesetting them directly. He wanted to typeset them all in a consistent font, rather than mixing glyphs from Computer Modern and other sources.

The first version Chuck sent me was in `.otf` format, with the characters we wanted (zero-slash, prohibition, etc.) replacing lowercase letters. So it sufficed to start up FontForge (by George Williams, `fontforge.sf.net`) and use its 'Generate Fonts' feature to create a `.pfb` + `.afm`, which takes the first 256 characters. Easy. (I wanted to use Type 1 since this was happening quite far along in the article's processing, and I had been using pdfLaTeX thus far; switching to XeLaTeX or LuaLaTeX would have meant losing functionality from `microtype` and thus losing considerable time fixing line breaks.)

Then Chuck sent me a revised font with additional characters. This time it was a `.ttf`, and the characters were in the correct Unicode positions (which are far beyond the first 256 characters, of course), so I couldn't just use the simple FontForge generation. (I could have asked Chuck to rearrange the characters, but I decided to take it as a challenge; after all, it's an article of our faith that TEX should be able to use any font.)

Instead, I followed the article by Hàn Thế Thành about using TrueType fonts directly in pdfTEX (30:1, `tug.org/TUGboat/tb30-1/tb94thanh.pdf`). First I created a custom encoding file, `altzero.enc`, starting like this:

```
/enclucidaaltzero [
  /emptyset     % U+2205
  /uni20E0      % prohibition
  /emptyset.var % glyph index #2225
  ... ]
```

These character names are specified in the font. I discovered them by looking at the font in FontForge and using 'View → Goto' to navigate to the characters; thankfully, searching for `uni...` works even when the character does not have a name of that form. Chuck told me the name of the variant emptyset glyph (zero-slash in this case), which does not have a Unicode assignment.

Still following Thành's article, I then made the `.tfm`:

```
ttf2afm -e altzero.enc -o altzero.afm ZeroFont.ttf
afm2tfm altzero.afm
```

In the LaTeX document, the font was used like this:

```
\pdfmapline{+altzero ZeroFont <altzero.enc
                               <ZeroFont.ttf}
\font\altzero = altzero
{\altzero\char0}% of our encoding: emptyset
```

All was fine, until Chuck sent me one more revision of the font. This time it was again `.otf`, but now using the Unicode positions. pdfTEX cannot read `.otf`, and converting `.otf` to `.ttf` seemed fraught with potential problems to me. So I used a third tool: `otftotfm` (by Eddie Kohler, `lcdf.org`). Once I read the documentation enough times, I happily discovered that I could re-use the same encoding file. The invocation this time:

```
otftotfm -e altzero.enc --no-encoding \
        ZeroFontOT.otf altzero
```

(The `--no-encoding` option just tells `otftotfm` not to generate its own new encoding file; the final `altzero` argument is the base name of the `.tfm` and `.pfb` generated by `otftotfm`.)

Usage in the LaTeX source is similar to the above, but now we have a `.pfb`:

```
\pdfmapline{+altzero ZeroFontOT <altzero.enc
                                 <ZeroFontOT.pfb}
```

The tools themselves output the map lines needed, according to the names embedded in the font files, etc.

Moving on from the technicalities, it was a great pleasure to work with Chuck on his articles in this issue. He has had a great (and positive!) influence on me, with recommendations for schools to attend, professors to work with, and personally encouraging my lifelong interest in typography and typesetting. As it turned out, we effectively finished work on the article on Chuck's birthday. Happy birthday Chuck!