Minimal setup for a (cyrillic) TrueType font

Oleg Parashchenko

Abstract

Our goal is to describe font installation in small steps. First, we typeset plain TEX text in the right encoding. Then, we describe the minimal setup to get the correct result in PDF using both the chain tex+dvips+ps2pdf and the direct pdftex, with notes on encodings and TrueType to Type 1 font conversion. One final step for LATEX is then given. The examples are based on a cyrillic font, but useful also for other scripts as well.

1 Introduction

Several sets of instructions on how to install fonts have already been written, among them: *The LATEX Companion* [6], *The LATEX Graphics Companion* [3], and fontinst documentation [2].

However, I didn't have luck with them. The first problem is that the installation process, as usually described, is "atomic": after magic spells, you get the font installed, with all the required entries in config files, all in one step. If something goes wrong, an inexperienced user doesn't have control points to check what is ok, and what went awry.

The second problem is that the tutorials use Type 1 fonts as the starting point. But when we are starting with a TrueType font, it is a good idea is to skip over that starting point, and use only a part of the standard PostScript way. Otherwise the process seems unclear and superfluous.

Recently I needed a cyrillic Helvetica for a document, the build system for which used the dvips chain. Facing the choice of re-implementing the system using XƎTEX or finding a way to install the font, I decided to try the latter once more.

My new approach was to throw away all the tutorials and instead move step by step on my own. First, typeset a text in the right encoding for plain TEX. Then get the text in DVI and its viewer xdvi. The next steps are PostScript and PDF. Finally, the font is integrated to LATEX's NFSS.

In this guide, all the font, config and test files are located in one directory. Putting them into the right places in the texmf tree is left as an exercise for the reader. One hint on that: To trace which files TEX is truly using, I found that instead of kpathsea debug options, it is more reliable and convenient to use the system utility strace.

As the font, I use here Helvetica Cyrillic from Linotype (font name HelveticaLTCYR-Roman, file name LT_51680.ttf). It is a proprietary font, but that doesn't matter for our purposes: there is nothing font-specific in this guide except the names.

2 Plain TEX source

Let's typeset the word привет ("hello" in Russian). The question is: which encoding to use? It's not important as long as both the text and the font use the same encoding. In the LATEX world, cyrillic is associated with T2A, so let's use it here too.

The next question is: what is the T2A encoding? I couldn't find a reference, therefore I copied the file t2a.enc from my texmf tree and studied it. The cyrillic letters are hidden behind the names afiiNNNNN. I didn't see any logic in the numbers, and therefore searched for documentation and found the "Adobe Standard Cyrillic Font Specification" [1].

Our word is encoded as afii10081 afii10082 afii10074 afii10067 afii10070 afii10084. After calculating the positions in the encoding vector (it turns out that the T2A codes are the same as in the Windows-1251 encoding, except for the letters Ë and ë), we are ready to typeset a test document plaintest.tex:

```
\font\f=lhcr8z  % error: unknown font
\f ^^ef^^f0^^e8^^e2^^e5^^f2\par
\bye
```

This file doesn't compile yet because TEX does not know the font lhcr8z (the name is explained later).

3 Font metrics

To compile a file, TEX doesn't need the fonts themselves, but only their metrics, which are stored in .tfm files. One way to get a .tfm from our TTF:

```
ttf2tfm LT_51680.ttf -T t2a.enc lhcr8z.tfm
```

The tool displays a number of warnings like "Cannot find character 'circumflex' specified in input encoding." Indeed, the font doesn't have a glyph named circumflex, but rather asciicircum. It is possible to define aliases, but for now I just ignored the warnings. The naming issue is a big topic, and an article [4] by Hàn Thế Thành explains it in detail and suggests a general solution. An alternative is to use fontforge instead of ttf2tfm.

Now running tex creates a .dvi file, and the log file is free of warnings: tex plaintest.tex.

4 xdvi and .pk font

However, running xdvi is not successful: xdvi requests the font, kpathsea finds no font and asks mktexpk to generate one, but mktexpk doesn't know how to create it. As the font lhcr8z is not found, xdvi uses the font cmr10 instead, but complains that the characters are not defined in it and displays an empty page. After consulting the documentation of `mktexpk`, I created a map file `ttfonts.map`:

```
lhcr8z  LT_51680.ttf  Encoding=t2a.enc
```

Now running `xdvi` automatically creates a `.pk` file, but it is stored in some cache directory. Therefore, I prefer to create `lhcr8z.600pk` explicitly (the resolution 600 came from the output of `mktexpk`):

```
ttf2pk lhcr8z 600
```

Now this works well and shows привет:

```
xdvi plaintest.dvi
```

5 testfont

As an optional step, it's useful to get the font table, which is also the encoding table. Process the file `testfont.tex` (TEX finds it automatically):

```
tex testfont.tex
```

It first asks for a font name (answer is `lhcr8z`) and then for commands. There are a number of them, but for our needs it is enough to say:

```
\table\bye
```

Look at the result: `xdvi testfont.dvi`.

To avoid font issues due to further experiments, convert the current result to a bitmap image:

```
dvipng -o testfont.png testfont.dvi
```

Check that the glyphs are located as expected and save the table for later reference. It will be interesting to compare the result with future output from `dvips` and `pdftex`.

6 From DVI to PostScript to PDF

The classical way to create PDF from TEX is to use `dvips` and `ps2pdf`. This already works for us, but the font inside is bitmap, not vector. To get the vector version, create a local `psfonts.map`, the default map file for `dvips`:

```
lhcr8z HelveticaLTCYR-Roman <lhcr8z.pfa
```

The font `lhcr8z.pfa` can be also used in its binary form, `lhcr8z.pfb`. In the next sections we will see how to convert from TTF to Type 1. After doing that, we are ready to produce the PDF:

```
dvips -o plaintest.ps plaintest.dvi
ps2pdf plaintest.ps plaintest.pdf
```

Now running `pdffonts` (from the `xpdf` package) reports that the font `HelveticaLTCYR-Roman` is embedded and its type is `1C`. A PDF viewer should welcome us with привет.

7 pdftex instead of tex+dvips+ps2pdf

One more tool, `pdftex`, one more map file is required. This time it is named `pdftex.map`. The content is one line (the line break here is editorial):

```
lhcr8z HelveticaLTCYR-Roman <t2a.enc
      <LT_51680.ttf
```

That's all we need; `pdftex` is ready to run:

```
pdftex plaintest.tex
```

8 Encodings: TTF to Type 1 conversion

After reading different font installation instructions, I was lost in details. What are all these files and do I really need them? Extensions `tfm`, `afm`, `pfa`, `pfb`, `vf`, `fd`, `enc`, `map`, the suffixes `8a` and `8r` for versions of fonts. Font re-encoding instructions in config files. For a cyrillic font, what are the equivalents for `8a`, `8r` and the magic spells?

Thanks to the step-by-step approach, the mess was soon localized into two questions: 1) how to convert a TrueType font to PostScript, and 2) how to name it for NFSS.

The first initially looked simple. A quick search pointed to the tool `ttf2pt1` [5], which supports different encodings, including cyrillic. But then there was a fight with technical troubles. First, due to some copyright protection trick, the result was unusable without the option `-a` (include all glyphs, even those not in the encoding table). Second, I falsely concluded that `dvips` required a virtual font to use the PostScript version. Third, I didn't specify font embedding in `ttfonts.map` (using the character `<`) and had to teach `HelveticaLTCYR-Roman` to `gs`/`ps2pdf`. But finally I got привет in PDF.

The naming issue was more tricky. The begining is obvious: `l` for Linotype, `hc` for Helvetica Cyrillic, `r` for regular. But what about the rest: `8a`, `8r` or something else? And meanwhile, for latin fonts, isn't only one of `8a` and `8r` required? (Answer: `8r` is enough.) The help came from an informal note in some tutorial: `8a` fonts should be re-encoded for use in TEX, `8r` fonts are ready to use in TEX.

Finally, the whole picture was clear for me. If a Type 1 font is made available to TEX as is, the NFSS name uses the suffix `8a`. After converting the font to TEX encoding, the suffix becomes `8r`. Further observations:

- PostScript fonts are not physically converted. Instead, they are re-encoded on the fly during PostScript execution. The corresponding commands are given through map files.
- With a TrueType font as the starting point, I don't see any reason to first convert TTF to Type 1 with the Adobe encoding, and then re-encode the font for use in TEX. Instead, I prefer to convert directly to TEX encoding.
- It seems there is a strong association between the suffix `8r` and T1 encoding, therefore for the cyrillic font I selected some other suffix, `8z`.

This idea of avoiding `8a` is obvious, but I was misguided by the help text of `ttf2pt1`. Among the supported encodings there is `adobestd`, which is commented as "`Adobe Standard, expected by TeX`" (wrong). Meanwhile, the encoding `cyrillic` seems to be `windows-1251`, not T2A. Investigations show that it is possible to get the correct result using map files (the option `-L`) and that the source code tarball of `ttf2pt1` contains maps for T1 and T2A, but these maps are not installed on my Linux. Therefore, currently it's inconvenient to use `ttf2pt1`. I'll submit a report to the developers, and hope they will improve the situation.

The alternative is the tool `fontforge` [8]. Initially I failed to convert the font correctly, but while fighting with `ttf2pt1`, I stumbled upon the documentation of the `comicsans` package [7], which described how to use `fontforge`. After adaptation to the current version, here are the instructions.

- First, teach `fontforge` about the T2A encoding. Click *Encoding→Load Encoding*, select the file `t2a.enc`.
- Select *Encoding→Reencode→T2AAdobeEncoding*. The glyphs are rearranged to the correct (for T2A) positions.
- Select *File→Generate Fonts*. You need to select options: PS Type 1 Ascii or Binary, No Bitmap Fonts, activate output of TFM and set Force glyph names to Adobe Glyph List.

9 LaTeX

The real problems are already solved. To integrate the font to LaTeX NFSS, one more config file is required, `t2alhc.fd` (the file name is font encoding name plus family name):

```
\ProvidesFile{t2alhc.fd}
\DeclareFontFamily{T2A}{lhc}{}
\DeclareFontShape{T2A}{lhc}{m}{n}
    { <-> lhcr8z}{}
```

A sample LaTeX document is shown next. Here `utf8` is used as the input encoding, conversion to T2A is done by LaTeX, thanks to the `inputenc` package. The package `fontenc`, among other useful actions, sets the default encoding for fonts to `T2A`, so `fontencoding` before `selectfont` is redundant and can be safely removed.

```
\documentclass{article}
\usepackage[T2A]{fontenc}
\usepackage[utf8]{inputenc}
\begin{document}
\fontencoding{T2A}\fontfamily{lhc}\selectfont
^^d0^^bf^^d1^^80^^d0^^b8^^d0^^b2^^d0^^b5^^d1^^82
\end{document}
```

Both `latex`+`dvips`+`ps2pdf` and `pdflatex` should produce the desired PDF.

10 Summary

A `.tfm` file is always required. Create it using either `ttf2tfm` or `fontforge`.

An encoding file is also always required. It can be found in your `texmf` tree.

Each tool consults its own map file for more information about the fonts. You need to provide the details (such as encoding) to get the correct result.

pdfTeX can use a TTF file directly.

`xdvi` and `dvips` use either `.pk` bitmaps or Type 1 outlines. A `.pk` font is created using `ttf2pk`.

A Type 1 font (and the corresponding `.tfm`) can be created using `fontforge`. The font ought to be created in TeX encoding. Do not use `ttf2pt1` yet, unless you understand what are you doing.

For integration in LaTeX NFSS, a `.fd` font description file is needed.

References

[1] Adobe Systems Incorporated. Adobe standard cyrillic font specification. See `http://www.adobe.com/devnet/font/pdfs/5013.Cyrillic_Font_Spec.pdf`.

[2] The fontinst home page. See `http://www.tug.org/applications/fontinst/`.

[3] Michel Goossens, Sebastian Rahtz, and Frank Mittelbach. *The LaTeX Graphics Companion*. Addison-Wesley, 1997.

[4] Hàn Thế Thành. A closer look at True Type fonts and pdfTeX. *TUGboat*, 30(1):32–34, November 2009.

[5] Mark Heath. TrueType font to PostScript Type 1 converter. See `http://ttf2pt1.sourceforge.net/`.

[6] Frank Mittelbach and Michel Goossens. *The LaTeX Companion*. Addison-Wesley, 2004.

[7] Scott Pakin. The comicsans package. See `http://www.ctan.org/macros/latex/contrib/comicsans/comicsans.pdf`.

[8] George Williams. Fontforge. See `http://fontforge.sourceforge.net/`.

◇ Oleg Parashchenko
bitplant.de GmbH
Fabrikstr. 15
89520 Heidenheim, Germany
`olpa (at) uucode dot com`