mf2outline

Linus Romer

March 2, 2015

Contents

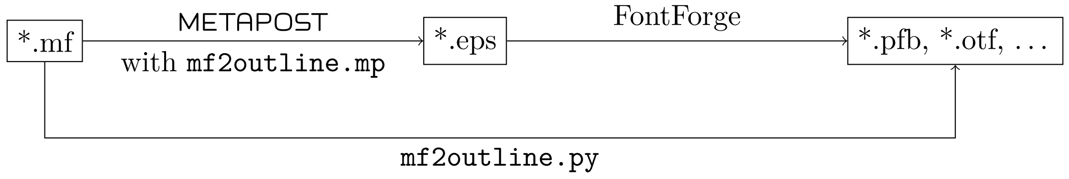
1 Introduction

METAFONT is a very versatile font description language, especially when you need to design several faces of a typeface family. However, the METAFONT compiler has some severe restrictions:

- The METAFONT compiler can only produce bitmaps and cannot produce outline font formats like Type 1 or OpenType.
- The METAFONT compiler cannot write more than 256 different characters per font.

Luckily, the METAPOST language and its compiler can be used as an expediant (see [Hobby13]). Together with the `mfplain.mp` base, the METAPOST language supersets nearly 100 % of the METAFONT language. The METAPOST compiler outputs PostScript files, which can be imported in FontForge and then be converted to outline font formats. This process is automated by the `mf2outline.py` script.

For compatiblity reasons, the `mfplain.mp` base does not support more than 256 different characters per font. To get over this and other artificial restriction, the `mf2outline.mp` base extends the capabilities of the `mfplain.mp` base. Of course, the backwards compatibility to METAFONT will be lost by using these extensions.



2 The `mf2outline.py` Script

2.1 Requirements

The following programs have to be installed before using `mf2outline`:

- Python interpreter (`mf2outline.py` is a Python script)
- METAPOST compiler
- FontForge's python extension (`python-fontforge`)

2.2 Usage and Command-line Options

The general usage for a METAFONT file `mfsource` is easy:

```
mf2outline.py mfsource
```

This will output an OpenType font file named `mfsource.otf` in your working directory. The file extension `.mf` of the specified METAFONT source file can be omitted.

You may add some of these optional arguments:

`-h, --help`
Show the help message and exit.

`-v, --verbose`
Explain what is being done.

`-vv, --veryverbose`
Explain very detailed what is being done.

`--designsize SIZE`
Force the designsize to be SIZE (e.g. 12 for 12pt).

`--raw`

Do not remove overlaps, round to int, add extrema, add hints…

`--preview`

Generate only the most important letters, use icosagon pens instead of circle/elliptic pens and do not care about advanced font features like kerning and ligatures (mainly used for METAFLOP).
List of letters: ! & ( ) , - . / 0 1 2 3 4 5 6 7 8 9 ? A B C D E F G H I J K L M N O P Q R S T U V W X Y Z a b c d e f g h i j k l m n o p q r s t u v w x y z

`-f FORMATS, --formats FORMATS`

Generate outline fonts in the formats FORMATS (comma separated list).
Supported formats: sfd, afm, pfa, pfb, otf, ttf, eoff, svg, tfm
Default: otf

`--encoding ENC`

Force the font encoding to be ENC.
Natively supported encodings: ot1, t1, unicode
Default: unicode
The file ENC.enc will be read if it exists in the same directory as the source file (the encoding name inside the encoding file must be named ENC, too).

`--fullname FULL`

Set the full name to FULL (with modifiers and possible spaces).

`--fontname NAME`

Set the font name to NAME (with modifiers and without spaces).

`--familyname FAM`

Set the font family name to FAM.

`--fullname-as-filename`

Use the fullname for the name of the output file.

`--fontversion VERS`

Set the version of the font to VERS.
Default: 001.001

`--copyright COPY`

Set the copyright notice of the font to COPY.

`--vendor VEND`

Set the vendor name of the font to VEND (limited to 4 characters).

`--weight WGT`

Force the OS/2 weight of the font to be WGT.
The weight number is mapped to the following PostScript weight names:

100	Thin
200	Extra-Light
300	Light
400	Book
500	Medium
600	Demi-Bold
700	Bold
800	Heavy
900	Black

`--width WDT`

Force the OS/2 width of the font to be WDT.
The width number stands for the following width names:

1	Ultra-condensed
2	Extra-condensed
3	Condensed
4	Semi-condensed
5	Medium (normal)
6	Semi-expanded
7	Expanded
8	Extra-expanded
9	Ultra-expanded

`--ffscript FFSCRIPT`

Specify an own finetuning fontforge script (e.g. finetune.pe). The script file has to be in the same directory as the source file. Example script:

```
Open($1);
SelectAll();
RemoveOverlap();
Generate($1);
Quit(0);
```

2.3 Restrictions

Not every valid METAFONT typeface can be automatically converted by `mf2outline`. The three most important restrictions are listed below:

- The METAFONT typeface cannot be compiled by METAPOST when it uses some special features of METAFONT that are not implemented in METAPOST (e.g. *Pandora*).

- If the font uses many overlapping filldrawn areas, FontForge does not always import the PostScript files correctly (e.g. Computer Modern). As a solution, you can use the `--raw` option and finetune the font by hand in FontForge.
- As a mathematical fact, a generic cubic beziér spline path that is drawn by a elliptic pen cannot be converted perfectly to cubic beziér spline outlines. Hence, FontForge does only an approximation job here. This approximation is normally very close to the original shape, but if you use heavily twisted cubic beziér splines, the approximation will be unsatisfactory.

2.4 METAFLOP

METAFLOP is an easy to use web application for modulating METAFONT fonts:

`http://www.metaflop.com/modulator`

The conversion to outline formats is being done by `mf2outline`.

2.5 Other Tools

The following two programs are alternatives to `mf2outline`.

`mftrace` is a python script that converts METAFONT fonts into Type 1 fonts. Unlike `mf2outline`, `mftrace` can cope with *every* valid METAFONT font. Unfortunately, the outline paths are not that neat.

`mf2pt1` is a perl script that converts METAFONT fonts into Type 1 fonts. Actually, `mf2pt1` is pretty similar to `mf2outline`, but does not rely that much on FontForge.

Both programs, `mftrace` and `mf2pt1`, have deeply inspired the author of `mf2outline`. Thus, many ideas of the two programs can be found in `mf2outline`, too.

3 The `mf2outline.mp` Base

The `mf2outline.mp` base extends the `mfplain.mp` base. Unlike `mf2outline.mp` base, `mf2outline.mp` causes METAPOST to write special additional glyph information to the PostScript files and to generate an additional file mf2outline.txt, that contains general font information. Normally, some of these additional information are stored in the `tfm` file.

There is a special version called `mf2outline-prev.mp` that causes METAPOST to use icosagon pens instead of circular/elliptic pens. The only difference to `mf2outline.mp` is the version number, that ends with the string `"prev"`.

The new extensions and its necessary macros are described in the following subsections.

3.1 Additional Font and Glyph Parameters

The `tfm` file stores amongst other things the following parameters:

- Global font parameters:
  - font_size
  - font_slant
  - font_normal_space
  - font_normal_stretch
  - font_normal_shrink
  - font_x_height
  - font_quad
  - font_extra_space
  - font_identifier (normally not stored)
  - font_coding_scheme (normally not stored)
- Glyph parameters:
  - charwd (character width)
  - charht (character height)
  - chardp (character depth)
  - charic (character italic correction)
  - charcode (code number of the character)
  - charext (code extension number of the character)
  - chardx (horizontal escapement of glyph positioning)
  - chardy (vertical escapement of glyph positioning)

The `mf2outline.mp` base defines some new parameters that cannot be stored in the `tfm` format:

- Global font parameters:
  - font_os_weight
  - font_os_width
  - font_version
  - font_copyright
  - font_name
  - font_fullname
  - font_familyname

- Glyph parameters:
  - charunicode (unicode string like "004A")

Most of these parameters are written to the font information file `mf2outline.txt` and read by `mf2outline.py`.

3.2 Unicode Support

3.3 Kerning

3.4 Ligatures

3.5 Other OpenType Features

References

[Hobby13] John D. Hobby et al. *METAPOST – A User's Manual*. `www.tug.org/docs/metapost/mpman.pdf`, 2013