Font Handling in Troff With PostScript Devices

FONT HANDLING IN TROFF WITH POSTSCRIPT DEVICES

Gunnar Ritter

10/24/06

Heirloom Documentation Tools

<http://n-t-roff.github.io/heirloom/doctools.html>

The basics

Heirloom *troff* understands two ways to select PostScript fonts.

The current method can access PostScript font files directly. Fonts are selected using an extended ".fp" request. As simple examples,

```
.fp 0 R somefont pfb
.ft R
Here is some text in somefont.
```
☞ Here is some text in somefont.

or, with the Linux Libertine font used in this document,

```
.fp 0 R LinLibertine_R otf
.ft R
Here is some text in Linux Libertine Regular.
```
☞ Here is some text in Linux Libertine Regular.

But it is also possible to have different names for the metrics and glyph data files, as in

```
.fp 0 R mykerning.afm somefont.pfb
.ft R
This text prints in somefont using modified kerning.
```
☞ This text prints in somefont using modified kerning.

The default higher-resolution "ps" PostScript device always uses AFM files; it supports the conventional ".fp" request for backwards compatibility to select pre-installed fonts from the PDF base set.

With fonts selected by this method, localized input processing is performed according to the LC_CTYPE environment variable, or according to a document-specific value set by the ".lc_ctype" request:

```
.\" Enable long request names.
.do xflag 3
```

```
.\" de_DE.utf8 is for GNU libc; de_DE.UTF-8 works elsewhere.
.lc_ctype de_DE.utf8
Deutscher Text mit ‹Anführungszeichen›
```
☞ Deutscher Text mit ‹Anführungszeichen›
```
.sp
The first five small letters of the Greek alphabet: α β γ δ ε
```
☞ The first five small letters of the Greek alphabet: α β γ δ ε

Note that any use of AFM, OpenType, or TrueType files requires that the output of *troff* is passed to the exactly matching version of *dpost,* i.e. to the one that was delivered with the same release of this package. Sending such *troff* output directly to a print spooler that invokes a system version of *dpost* will thus usually not work.

The historical method requires font tables to be generated manually. It is still supported, and is still the default for the lower-resolution "post" PostScript device, but it is not recommended anymore that it is used for adding fonts. Localized input processing according to LC_CTYPE is not available with this method.

Installing PostScript Type 1 fonts

Making additional fonts available is easy with the current method. If you buy a PostScript Type 1 font for use with *troff,* select the Windows version. You will get a set of files from the vendor. Only two of them are of interest for *troff:*

xxxxxxxx.afm
: This is the metrics file. *troff* accesses it to learn the widths of characters in the font.

xxxxxxxx.pfb
: This file contains the actual glyph data. *troff* does not really need this file, but the printer or PDF converter does. In many cases, it is most convenient to include the data directly into the .ps file generated, as in the examples above.

Create a directory to hold your fonts. For compatibility with future versions of *troff*, it is recommended that it is put outside the *troff* hierarchy /usr/ucblib/doctools; something below /usr/local may be a good choice. For each font you want to install, copy or link both the AFM and the PFB file into the directory. Set the TROFFONTS environment variable to this directory; if you have multiple font directories, you can separate them in TROFFONTS by colons, just as in the PATH variable for the shell, e.g. "TROFFONTS=/usr/local/share/fonts:/usr/share/fonts export TROFFONTS". Write the definition in your ".profile" or a similar startup file to make it permanent. You can now use the fonts with *troff*. If you want to install more fonts later, it is sufficient to put them into the font directory.

Installing OpenType or TrueType fonts

The procedure for OpenType and TrueType fonts is nearly the same as for Type 1 fonts, except that there is only one file "FontName.otf" (or "FontName.ttf") with them that contains both metrics and glyph data. Just copy this file to a directory given in the TROFFONTS path.

Using OpenType features

High-quality OpenType fonts may contain feature tables that allow special typographic effects. For example, the Linux Libertine font contains superscripts characters that can be accessed with the ".feature" request in *troff:*

```
.do xflag 3
.fp 1 R LinLibertine_R otf
.fp 0 U LinLibertine_R otf
.feature U +sups
.ft R
Regular text which ends with\fUsuperscript\fR.
```

☞ Regular text which ends with$^{\text{superscript}}$.

To retrieve a list of features in an individual font that are available with *troff*, use "otf_info –s font.otf". Typical useful features are:

`+c2sc`	converts capitals to small capitals
`+case`	substitutions for use in combination with text in all-capital letters
`+onum`	old-style numerals
`+pnum`	proportional (lining) numerals
`+pnum +onum`	proportional old-style numerals
`+smcp`	converts lower-case letters to small capitals
`+smcp +c2sc`	converts all letters to small capitals
`+titl`	titling characters
`+sups`	superscript
`+infs`	subscript
`+zero`	slashed zero

Features such as "aalt" or "ornm" which only map sets of very special characters to different positions are normally not useful with *troff* since these characters can be accessed directly using "\[name]" escape sequences anyway. To make an individual alternate character the default, the ".ftr" request can be used:

```
.do xflag 3
.fp 0 R LinLibertine_R otf
.ftr R h\[h.alt]
.ft R
Using alternate character in the word ``hello.''
```

☞ Using alternate character in the word ``ɦello.''

Feature mappings result in changes to the same internal data structures as the ".ftr" request. It is thus possible to make adjustments to mappings by using ".ftr" after ".feature", or to create personalized variants by using ".ftr" based on the data obtained from "otf_info –s".

The ".feature" request must still be active for a font when a character is printed; disabling a feature with ".feature F –feat" only works completely if all

of its mappings are still in position. It is thus recommended that the ".feature" request is used only once for a font immediately after it has been mounted, and that a font is mounted multiple times, each time for enabling an individual feature set. Doing so has the additional advantage that switching between features is conveniently possible using the "\f" escape sequence.

Using fonts with old-style numerals

A Type 1 font with old-style numerals but otherwise standard characters usually does not require any special mechanism. It can simply be mounted, selected, and used.

Old-style numerals contained in OpenType fonts can be accessed conveniently by mapping them to the standard ASCII numbers with the ".feature" request:

```
.do xflag 3
.fp 1 R LinLibertine_R otf
.feature R +onum
The numerals 0 1 2 3 4 5 6 7 8 9 are in old-style.
```

☞ The numerals 0 1 2 3 4 5 6 7 8 9 are in old-style.

Using fonts with small capital letters

Small capital fonts sometimes need letter space tracking. Thus e.g. to use the Linux Libertine small capital font, you can write something like

```
.do xflag 3
.fp 0 SC LinLibertine_R otf
.feature SC +smcp
.track SC 1 1.2 40 0
.ft SC
This text prints in Small Capitals.
```

☞ THIS TEXT PRINTS IN SMALL CAPITALS.

Using a combination of expert and standard fonts for small capitals

A Type 1 expert font contains small capital letters but no upper-case capital letters. Since it would be very inconvenient to change the font explicitly for each upper-case letter, the fallback sequence is useful in this case. The expert font is selected as text font, but for each upper-case letter encountered, *troff* changes automatically to the standard font:

```
.do xflag 3
.fp 1 R normalfont pfb
.fp 0 E expertfont pfb
.track E 1 .2 24 2
.fallback E R
.ft E
This text prints in Small Capitals.
```
☞ THIS TEXT PRINTS IN SMALL CAPITALS.

For OpenType fonts, all small capital letters are contained within the base font, and the "smcp" feature accesses them as shown above.

Using a combination of expert and standard fonts for old-style numerals

This also uses the fallback sequence, but since the standard numerals are present in the standard font, they have to be hidden first so that *troff* does not select them:

```
.do xflag 3
.fp 1 R normalfont pfb
.fp 0 E expertfont pfb
.fallback R E
.hidechar R 0123456789
.ft R
The numerals 0 1 2 3 4 5 6 7 8 9 are in old-style.
```
☞ The numerals 0 1 2 3 4 5 6 7 8 9 are in old-style.

Using the expert font for both small capitals and old-style numerals

The examples above can also be combined. Since the ".track" request does not affect the tracking of fonts that are selected by the fallback sequence, it is sufficient to mount each font once:

```
.do xflag 3
.fp 1 R normalfont pfb
.fp 0 E expertfont pfb
.fallback R E
.fallback E R
.track E 1 .2 24 2
.hidechar R 0123456789
.ft R
The numerals 0 1 2 3 4 5 6 7 8 9 are in old-style.
```

☞ The numerals 0 1 2 3 4 5 6 7 8 9 are in old-style.

```
.br
.ft E
But this text prints in Small Capitals.
```

☞ But this text prints in Small Capitals.

Using the expert font for ligatures

PostScript Type 1 fonts usually do not contain ligatures for ff, ffi, and ffl in the base font; an expert font delivers them in addition. Automatic substitution of such ligatures can be enabled using the ''.flig'' request in combination with the fallback sequence:

```
.do xflag 3
.fp 1 R normalfont pfb
.fp 0 E expertfont pfb
.fallback R E
.flig R ff \(ff ffi \(Fi ffl \(Fl
.ft R
```

```
effective office offline
```
☞ effective office offline

OpenType fonts contain all available ligatures in the base font, so no special procedure needs to be applied with them.

Defining additional ligatures

Fonts may provide ligatures beyond the five standard *troff* ones. They are not activated by default regardless of any definitions in the font metrics files, but can be enabled individually as desired; *troff* will then replace sequences of characters automatically with them as with the standard ligatures. For example, Linux Libertine contains ligatures for use with the historic "ſ" letter variant:

```
.do xflag 3
.fp 1 R LinLibertine_R otf
.flig R \[longs]\[longs] \[longs_longs] \[longs]i \[longs_i]
.flig R \[longs]\[longs]i \[longs_longs_i]
U\[longs]e the\[longs]e ligatures
for cla\[longs]\[longs]ic typography.
```
☞ Uſe theſe ligatures for claſſic typography.

troff splits ligatures in two parts when hyphenating words. It is thus necessary that any part of a ligature that is not a single character is also defined as a ligature, so "ﬀ" and "ﬁ" are prerequisites for "ﬃ".

It is normally not advisable to enable ligatures with special display forms (such as in "sec͡tion") using the ".flig" mechanism; they should be set manually as special characters ("\[c_t]" in this case) at precise locations.

Choosing between multiple possible ligatures

If a font contains only two-character ligatures, there are two possibilities for automatic ligature building with an input sequence that would form a three-

character ligature, e.g., "ffi" could be constructed as "ff i" or "f fi". *troff* normally selects the first ligature available, so "ff i" would be the default. In many cases, the other choice would be aesthetically preferable; this is the application for the ".fdeferlig" request:

```
.do xflag 3
.fp 1 R LinLibertine_R otf
.flig R \[longs]\[longs] \[longs_longs] \[longs]i \[longs_i]
.fdeferlig R \[longs]\[longs]i
Choo\[longs]ing the \[longs]econd ligature looks po\[longs]\[longs]ibly better.
```

☞ Chooſing the ſecond ligature looks poſſibly better.

Combining different fonts

Different fonts often have different visual sizes even if they are set in the same nominal point size. Also fonts from different vendors are often based on different standards for character heights. To adjust this, the ".fzoom" request is available. As implied by this purpose, it only applies to characters that are actually in the current font, not to characters from another font that have been selected using the fallback sequence.

```
.do xflag 3
.fp 1 R LinLibertine_R otf
.fp 0 M LinLibertine_M otf
.track M 1 -.1 40 -0
.fzoom M .8
\fMLibertine Monospace\fR must be adjusted to fit with Roman.
```

☞ Libertine Monospace must be adjusted to fit with Roman.

The ".fzoom" request affects all characters set in the respective font on the current output line. To create single zoomed words, mount a font twice under different names, but zoom it only once.

Spacing out individual words

Spacing out the characters of individual words is commonly done for highlighting text e.g. with German blackletter faces; the ".track" request can be used for this as well. Like ".fzoom", ".track" applies to all characters in the respective font on an entire output line. Thus it is also necessary to mount a font twice but to track it only once when it is used for tracking individual words.

Setting text in all-capital letters

When setting text in all-capital letters, it is normally necessary to zoom the font to a smaller size than that of the lower-case text surrounding it, and to track it as well. Mounting the regular font a second time is the most convenient way to handle this:

```
.do xflag 3
.fp 1 R LinLibertine_R otf
.fp 0 C LinLibertine_R otf
.track C 1 1.2 40 0
fzoom C .9
Regular text and \fCALL-CAPITAL TEXT\fR which ends.
```

☞ Regular text and ALL-CAPITAL TEXT which ends.

Pairwise kerning

Pairwise kerning is enabled unless the –x0 option is given. The ".kern" request disables it. Kerning tables are initially read from the AFM, OpenType, or TrueType files; this default kerning only applies if two adjacent characters are from the same font. A font-specific kerning table can be disabled using the ".fkern" request.

There are two methods to adjust the kerning tables: In some cases, it is most convenient to create a private copy of the AFM file and to adjust the "KPX" entries in it. This has the advantage that the modified kerning pairs are imme-

diately available for use in other documents.

For OpenType or TrueType fonts, or for cases where this is not suitable with Type 1 fonts, or not possible because the characters in the kerning pair originate from different fonts, *troff* provides the ".kernpair" request:

```
.do xflag 3
.fp 1 R LinLibertine_R otf
.feature R +onum
.kernpair R 0 R / 50
.kernpair R / R 1 -90
.kernpair R 1 R / 40
.kernpair R / R 2 -80
0/1/2
```

☞ 0/1/2

In this example, old-style numerals from Linux Libertine are mixed with the slash character, and are adjusted to achieve matching visual letter spacing.

Kerning of a character in any combination

It is sometimes useful to add a certain amount of space whenever a character appears, for example before ";" or "?", or on the inner sides of guillemots in French/Swiss style. The ".kernafter" and ".kernbefore" requests are useful for this:

```
.do xflag 3
.kernafter R ; 66 ? 66
```

If the characters affected by these requests are also member of a regular kerning pair, the resulting added space is the sum of both definitions. These requests are applied at the same places as kerning pairs, i.e. they have no effect if following or preceding a motion command, "\&" or line margin, and they are additionally restricted to have no effect if the other character is a space.

Hanging characters

Since there are no tables for hanging characters in AFM files, values must be given explicitly in *troff* source code using the ".lhang" and ".rhang" requests. Both accept a font specification followed by one or more pairs of characters and values:

```
.do xflag 3
.ps 10
.fp 1 R somefont pfb
.lhang R V -50 J -40
.rhang R \(hy 80
```

Adjustments are given in units of 1/72 000 inch multiplied by the actual point size, or 1/1 000 of the em size when the line is printed. Thus in this example, the left margin is shifted to the left by .5 points when the leftmost letter is a "V", and by .4 points when the leftmost letter is a "J"; the right margin is shifted to the right by .8 points when the rightmost letter is a hyphen.

Left margin adjustments are evaluated before the letters that fit on the current line are computed, and can thus principally be of any length. In contrast, right margin adjustments are evaluated after this computation is finished, and the adjustment is simply added to the word space of the output line. Thus a positive right adjustment that is large in relation to the line length will cause visible holes, and a negative adjustment will ultimately cause the words on the line to be printed over each other. This is not a problem for the typical application of hanging punctuation for visual alignment, though; if a line with eight word spaces is shifted in the example above, each word space is enlarged by only 1/100 em.

Mathematical and other special characters

Special mathematical characters like "\(+-", greek letters like "\(*a", and, with the "pslow" device, even the punctuation characters \(or \– \` \´ " # < > @ \ ^ ~ are normally not chosen from the current font, but are taken from the spe-

cial font instead. (PostScript names (e.g. "\[numbersign]", "\[less]", "\[at]") access characters from the current font.) The ".fps" request can be used to override this behavior. In particular, this is useful to set mathematical text with a different font. To use Linux Libertine for greek letters and mathematical symbols:

```
.do xflag 3
.fps math,greek,punct,large 1 R LinLibertine_R otf
.fp 2 I LinLibertine_RI otf
.fallback I R
.EQ
a ^=^ pi {r sup 2}
.EN
```

Helper utilities

The *otf_info* utility shows the contents of an OpenType or TrueType font file just as *troff* interprets it. It emits an ASCII format that is readable by humans and can also be used for further processing with shell scripts. It is useful to retrieve lists of characters and features supported with the font.

The shell script "stuff/showfont.sh" in the source code distribution uses *troff* and possibly *otf_info* to print a map of all characters in an AFM, OpenType, or TrueType font along with their PostScript names.

Limitations

dpost uses a method to embed CFF-based (PostScript-style) OpenType fonts in PostScript documents that is only available with PostScript 3 interpreters; older printing equipment that uses PostScript Level 2 or below cannot handle such documents directly. The recommended workaround is to convert the PostScript output to a PDF document and to print it using a PDF viewer or reverse conversion program. Also you may have to update your copy of Ghostscript in order to create PDF files with proper font embedding from such output.

Embedding TrueType fonts in PostScript documents requires PostScript 3 or PostScript Level 2 of at least version 2013.

troff only supports OpenType features that result in single-character substitutions insensitive of context (except for the "fi fl ff ffi ffl" ligatures and kerning which are enabled by default if possible). Also *troff* ignores any features that are not mapped to the "DFLT" or "latn" languages in an OpenType font.