Fonts

The installation and use of OpenType fonts in LaTeX

John D. Owens

Abstract

The emerging file standard in digital typography is the OpenType font standard, jointly developed by Microsoft and Adobe. OpenType fonts are natively supported by several popular operating systems and have many features and advantages that make them desirable for high-quality typography. However, OpenType fonts are not natively supported by the standard TeX engine. This article is a practical guide to installing OpenType fonts for use as text fonts in LaTeX.

The steps to install an OpenType font for use in LaTeX are:

1. For each OpenType font file, and for each combination of attributes for that font file, generate and install font metric and encoding files.
2. Next, for each font family, generate and install a font description (.fd) file that maps LaTeX font selection commands to the installed font files.
3. Finally, write and install a style (.sty) file that allows the user to select the font and its options for use within TeX.

We begin by discussing font background, the TeX font handling scheme, and existing font tools, then describe each of the three steps above in detail.

1 Font basics and font families

The advanced typographic features of the OpenType font format have motivated its widespread use in a variety of demanding applications. Before we dive into supporting OpenType in TeX, however, let's take a step back and look at our eventual goal. As a TeX user, we are less interested in using just a single font with a single set of options and more interested in using a *font family*: a collection of compatible font variants, usually from the same typeface, that can be used together. For instance, we might want to group together a plain and an italic form of a particular typeface into a family. We might want to make small caps available in our family as well, or perhaps incorporate several different weights or optical sizes. Once we have defined our font family, we would then like to ask TeX to enable the entire family with a single command. As an example, this document is typeset in an Adobe OpenType Minion Pro font family with old-style figures, with code segments typeset in Adobe's OpenType Myriad Pro, using the following LaTeX commands:

```
\usepackage{minion}
\usepackage[tt,sf,lining,scale=0.92]{myriad}
```

What are the different typeface alternatives that can be part of our font family? (Gelderman provides an introduction to typeface characteristics [3].) We can group possible alternatives into several broad categories, and then indicate how OpenType handles each category.

In different files	Within one OpenType file
weight (light, **black**)	kerning (VAVAV vs. VAVAV)
width (abc vs. abc)	ligatures (fi vs. fi)
optical size	figure style (1234 vs. 1234)
variant (e.g. *italics*)	Small Caps

Table 1: Font features provided by different OpenType files (left) and within a single OpenType file (right).

Our first four categories are weight, width, optical size, and variant. The *weight* of a typeface refers to the thickness of the strokes that constitute its glyphs. A font designer may also vary a typeface's *width* relative to its height. The major advantage of vector font formats (such as OpenType, PostScript Type 1, and TrueType) is their ability to be scaled to any size. However, type designers have found that the most visually appealing fonts are ones that are designed for a particular size range, called an *optical size*. Finally, the standard upright "roman" style for fonts is not the only possible style. Font users may also desire italic or oblique or outline forms of a particular typeface, which together are termed *variants*. In LaTeX's New Font Selection Scheme (NFSS), the combination of weight and width is called *series* and the variant is called *shape*; we use this terminology in Section 4.2.

In OpenType, each unique combination of weight, width, optical size, and variant is associated with a separate font file. As an example, the Adobe Kepler typeface has several alternatives in each of these categories. Kepler features six weights (light, regular, medium, semibold, bold, and black), each with four widths (condensed, semicondensed, regular, and extended). Most of Kepler's combinations of weight and width feature four optical sizes (from smallest to largest, caption, regular, subhead, and display), and each weight-width-optical-size combination has both an upright and italic variant. Thus it is little wonder that Kepler's many combinations require 168 different OpenType files. And after we catch our breath to consider all the typographical options already available to us, we dive into a single OpenType file to find still more options.

In addition to the categories that require different files, the OpenType font format also allows a single font

Editor's note: Due to the nature of this article, it is typeset in the Adobe Minion and Adobe Myriad typefaces. We thank Adobe for permission to use these fonts in both the print and web publications.

file to specify a variety of other features. Not all of these features are currently supported in TEX, but many of them are. For instance, the *kerning* feature adjusts the spaces between pairs of glyphs. Enabling *ligatures* replaces pairs of glyphs like 'f' and 'i' with a single 'fi' glyph. Besides kerning and ligatures, the two classes of OpenType features that we cover in this article are the choice of figure styles (for example, old-style [01234] vs. lining [01234]) and SMALL CAPS.

Table 1 summarizes the features provided by different OpenType files and within an OpenType file. With this overview of the many typeface alternatives that we would like to assemble into families, we can turn to how TEX interacts with fonts.

2 TEX font handling

In modern operating systems such as Microsoft Windows and Apple Mac OS X, applications that use OpenType fonts can read font information directly from the OpenType file. TEX, on the other hand, stores font information in a variety of files, and the complexity of creating and installing these files is the major reason that font handling has traditionally been a tricky task in TEX.

The OpenType font format can contain font data in either of two formats, Adobe PostScript Type 1 or TrueType. In this section we describe the font files that are associated with Type 1 and Type-1-flavored OpenType fonts. To support a Type 1 font, TEX requires the following files, each with its own function, with file locations specified by the TEX Directory Structure standard. More detailed descriptions of these formats can be found in Alan Hoenig's *TEX Unbound* [5] and Chapter 7 of the second edition of *The LATEX Companion* [11].

TFM "TEX Font Metrics" (tfm) files describe the dimensions of each character (glyph), along with a few font-wide parameter, which together are used by TEX to perform layout.

PFB For Type 1 fonts, "Printer Font Binary" (pfb) files contain Adobe PostScript Type 1 procedures that describe the shape of each glyph. These procedures are included by the output driver (for example, the dvips or pdftex program) in the output file (for example, PostScript or PDF).

VF "Virtual Font" files provide a mapping between the glyphs in the tfm files and the glyph order used by TEX (which is, in turn, specified with the encoding file, below). They are not needed for all fonts.

ENC Encoding files specify an ordering of glyphs called the "font-encoding vector". While typeset documents in English might require only the glyphs in TEX's default "OT1" font-encoding vector (used by Computer Modern Roman, for example), other languages or scripts need more or different glyphs. In this article, we use the "LY1" encoding, an alternative to OT1 developed by Y&Y that is well-suited for Type-1-flavored fonts. (Among other advantages, the LY1 encoding maps directly to Adobe's font encoding and thus requires no virtual fonts.)

MAP Finally, the map files tie the above files together. map files (and map file formats) are specific to output drivers and associate tfm and Type 1 font names with pfb files, which contain the shapes of glyphs in those fonts.

Only when these files have been properly installed for a particular font, and system databases updated, can TEX then typeset glyphs from that font in a document.

Writing all these files by hand would be both tedious and error-prone, so two excellent pieces of software have automated the font installation process.

- fontinst [6], by Alan Jeffrey, Rowland McDonnell, and Lars Hellström, automates the installation of PostScript Type 1 fonts into TEX. Philipp Lehman's font installation guide [9] is an outstanding tutorial for fontinst.

  However, fontinst does not offer access to OpenType features. Also, fontinst scripts are written in TEX, and are challenging for non-experts to write and use.
- Eddie Kohler's otftotfm [8], part of his LCDF Typetools suite, creates and installs the required TEX files (tfm, pfb, vf, enc, and map) from OpenType font files. Note that otftotfm generates PostScript Type 1 fonts from OpenType; please ensure that the legal license for your fonts allows such a format conversion. otftotfm is a command-line tool that accepts a set of options and applies them to a single OpenType font file.

  Section 6 describes two tools built around otftotfm that both automate calls to otftotfm across multiple font options and also create the necessary TEX fd and sty support files.

In this article, we focus on otftotfm as the underlying tool that translates OpenType fonts into a TEX-readable form. We also focus on the procedure for setting up text fonts. The setup for math fonts requires additional commands described in Chapter 7.10.7 of *The LATEX Companion* [11]. The next section outlines how otftotfm installs OpenType fonts, and the remainder of the article describes how to extend otftotfm to handle multiple font files and font families and how to make the installed fonts available to TEX users.

3 OpenType to TEX

The first step in making OpenType fonts available to TEX users is to deposit the various font files into the TEX installation for each variant in the font family. We begin by

showing how otftotfm installs a single font, using Adobe Minion Pro's Semibold Italic font as an example.

```
otftotfm -a -e texnansx -fonum -fkern -fliga \
  MinionPro-SemiboldIt.otf \
  LY1-MinionPro-SemiboldIt-onum
```

Let's analyze this example. -a is the magic "automatic" flag, automatically installing the relevant TEX font files from Section 2 (tfm, pfb, vf, enc, and map) into their proper locations within the TEX directories. -e texnansx specifies the encoding file for the LY1 encoding. Three OpenType features (old-style numerals, kerning, and ligatures) are requested with the -f flags, and the final two arguments are the names of the OpenType input font file and the output font name. The otftotfm manual explains these options in detail, and also enumerates available OpenType features [8].

Extending otftotfm to more input fonts and more variants is straightforward: simply call otftotfm for each and every combination of desired features. For complex variant combinations and fully featured font families, the number of calls to otftotfm can exceed many hundreds. The tools described in Section 6 automate this process.

LCDF's otfinfo tool [8] can identify the supported OpenType features for any OpenType font file, but which features are interesting for TEX users?

- The kerning (kern) and ligature (liga) features should always be turned on if available.
- OpenType fonts may support several kinds of numerals; onum (old-style numerals) and lnum (lining numerals) can both be supported in TEX and are commonly requested typographic features.
- Small Caps are enabled by the smcp feature.
- Superior (sups) and inferior (sinf) figures are useful for footnotes, inline fractions, and scientific typesetting; swashes (swsh) are more *DECORATIVE* alternatives to standard characters.

The otftotfm web page [8], in "otftotfm examples", contains examples of more advanced OpenType features, but as we note in Section 7, more advanced features rarely have high-level support in TEX or LATEX. In this article we concentrate on the overall installation procedure for OpenType fonts and support of more basic features; readers in need of more advanced features may consider the ConTEXt environment [10] or XETEX [7].

After otftotfm finishes installing all font files into TEX, texhash and updmap must be called to make TEX aware of the new installation. Now, the new fonts are available for typesetting in TEX — but how? The next section describes how to instruct TEX to *use* the correct font. TEX uses the "font description" file for this purpose.

4 Font description (fd) files

A font description file is a TEX source file that maps installed font file names to font attributes as used in (LA)TEX. Typically, each font family is described by a single fd file. As we previously mentioned, these techniques are applicable to text fonts; math fonts require additional commands [11].

Only two TEX commands are necessary in an fd file. The first declares a font family, and the second declares a specific font within that font family. We'll look at them one at a time.

4.1 \DeclareFontFamily

The \DeclareFontFamily command indicates that a certain font family is available in a certain encoding scheme. The names of encoding schemes are fixed (as mentioned before, we use LY1 in this paper), but the name of the font family is arbitrary. The most well-known naming scheme is Karl Berry's fontname scheme [1], which concatenates a unique three-letter code for each typeface with a one-letter suffix that indicates the font family (Section 4.4). (This naming scheme is required when using nfssext.sty, described in the next section.)

Let us continue with the Minion-Pro-with-old-style-numerals example; Minion is abbreviated pmn, and font families associated with old-style numerals are designated by j (more details about what constitutes a font family are in Section 4.4). The resulting command is:

```
\DeclareFontFamily{LY1}{pmnj}{}
```

The third argument to \DeclareFontFamily is less often used; it can contain special options for font loading and is explained in *The LATEX Companion* [11].

4.2 \DeclareFontShape

The \DeclareFontShape command associates a particular font with a particular combination of encoding, font family, series, and shape, a classification which we previously discussed in Section 1. To classify the particular Adobe Minion font we installed in Section 3, we invoke the following 6-argument command:

```
\DeclareFontShape{LY1}{pmnj}{sb}{it}{
  <-> LY1-MinionPro-SemiboldIt-onum}{}
```

The first four arguments are the classification; the fifth argument is the font file(s) associated with that classification; and the last argument is used in a similar way to the third argument of \DeclareFontFamily. This particular command associates the combination of LY1 encoding, Minion-with-old-style-numerals font family, semibold series (sb), and italic shape (it) with the installed font named LY1-MinionPro-SemiboldIt-onum. Note this font name is (necessarily) identical to the output name in the command we invoked in Section 3.

With only these two commands, you can specify a completely functional fd file. Three additional commands are useful, however, for a more fully featured family.

Optical size variants Now, what's the <-> symbol before the font name (in the above \DeclareFontShape example)? It's the size range and indicates the font sizes associated with that font name. <-> is actually a special case of the more general notation <n-m>, meaning "use this font only for point sizes greater than or equal to n and up to m". Removal of n or m removes the bound, so <-> indicates a match for all font sizes. With this notation, the extension to multiple font files for a particular combination at different sizes (necessary for optical size variants) is straightforward:

```
\DeclareFontShape{LY1}{pmnj}{sb}{it}{
 <6-8.4>  LY1-MinionPro-SemiboldItCapt-onum
 <8.4-13> LY1-MinionPro-SemiboldIt-onum
 <13-19.9> LY1-MinionPro-SemiboldItSubh-onum
 <19.9-72> LY1-MinionPro-SemiboldItDisp-onum}{}
```

Font substitution What happens if you're missing a particular variant for a font family? The sub command allows the substitution of one variant for another. For instance, few font families feature a slanted (oblique) variant, so fd files often substitute italic for slanted if slanted is requested. The following command asks for any reference, at any size, to semibold-slanted in our font family to be satisfied instead by semibold-italic.

```
\DeclareFontShape{LY1}{pmnj}{sb}{sl}{
 <-> sub * pmnj/sb/it}{}
```

Besides substituting italic for slanted, substituting bold for bold-extended is also common, as in the example below for the normal (n) shape.

```
\DeclareFontShape{LY1}{pmnj}{bx}{n}{
 <-> sub * pmnj/b/n}{}
```

Scaling Finally, \DeclareFontShape permits a font to be automatically scaled through the size command,[1] which is invoked by placing the scaling factor in brackets between the size range and the filename. The example below instructs TEX to typeset Minion's semibold italic variant at 95% of its natural size.

```
\DeclareFontShape{LY1}{pmnj}{sb}{it}{
 <-> [0.95] LY1-MinionPro-SemiboldIt-onum}{}
```

4.3 Naming shape and series

The de facto standard for the abbreviation strings associated with shape and series in fd files is described by LATEX's "New Font Selection Scheme" (NFSS) [15]. Any choice for shape and series abbreviation strings, including NFSS, must work together with the font selection commands in Section 5. Philipp Lehman's tutorial contains a fairly complete mapping between weight, width, and variant names and NFSS encodings [9]; we reproduce common encodings in Table 2.

Weight	NFSS series
light	l
book	m
regular	m
medium	mb
demibold	db
semibold	sb
bold	b
black	eb
Width	**NFSS series**
condensed	c
narrow	n
semicondensed	sc
regular	—
semiextended	sx
extended	x
Variant	**NFSS shape**
normal (upright)	n
italic	it
slanted	sl
oblique	sl
outline	ol
small caps	sc
small caps + italic	si

Table 2: A selection of NFSS codes for font weights, widths, and variants. From Lehman [9].

Lehman presents the following algorithms for generating the series and shape abbreviations used in \DeclareFontShape. First, the weight and width are combined to create "series". If both weight and width are "regular", the series is set to m; otherwise the series is set to the concatenation of the weight and width codes, ignoring "regular" if present. Creating the shape is also straightforward: if the variant is "regular", the shape is n, otherwise the shape is the concatenation of all variant codes, with the exception of small-caps and italics. This shape is instead designated si, and font selection using si is described in Section 5.3.2.

4.4 Font families

Some font features do not fit into the series-shape scheme. The most common of these features is numerical figure types, which may vary as lining (1234), old-style (1234), superior (^{1234}), inferior ($_{1234}$), and so on. To handle font selection with different styles of figures in TEX, typically,

[1] This is most common when two different typefaces that do not match in size are used together in a document; in the next section we expose this capability to the document author.

each type of figures generates its own font family. To generate the name of the font family, the three-letter font designation has a one-letter suffix appended to its 3-letter font name. Lining figures are associated with no suffix or with x, the "expert" suffix; old-style figures are j; superiors receive 1 and inferiors 0; and so on. Thus the Minion (pmn) font family with lining figures is pmnx; Minion with old-style figures is pmnj; and so on. Section 5.3.3 shows how to perform font selection with different font families.

5 Style files

At this point we have installed our fonts into TEX (Section 3) and categorized them by family, shape, and series (Section 4). The final step is to make those fonts available to the TEX document author as text fonts. The tools described in Section 6 automate the creation of the sty files that contain the commands in this section.

5.1 Selecting a font family

The default "roman" (text) font family is defined by the TEX variable \rmdefault. Redefining \rmdefault to another font family (as named by \DeclareFontFamily) resets the roman font family. For instance, the command below sets the current font family to our example font family, Adobe Minion with old-style figures.

```
\renewcommand*{\rmdefault}{pmnj}
```

In fact this is all we need to do to use our new font family. (Similarly, we set the default sans serif font family by setting the variable \sfdefault, and the typewriter family with \ttdefault.) However, rather than using one of these commands directly in TEX files, it's typical to instead wrap it in a style file and invoke \usepackage on that style file to perform this declaration. A minimal (but complete) style file called minion.sty for LATEX 2ε that uses the LY1 encoding follows.

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{minion}[Minion Pro OSF v0.99a 9/06]
\RequirePackage[LY1]{fontenc} % uses LY1 encoding
\renewcommand*{\rmdefault}{pmnj}
\endinput
```

5.2 Selecting between multiple font families

What if we'd like to use the same style file to support Minion font families with both old-style (pmnj) and lining (pmnx) figures? We use a package option to choose between the two font families:

```
\usepackage[oldstyle]{minion}
```

or

```
\usepackage[lining]{minion}
```

The extended style file that supports these options is:

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{minion}[Minion Pro OSF v0.99b 9/06]
\RequirePackage[LY1]{fontenc} % uses LY1 encoding
\DeclareOption{lining}{\renewcommand*{%
 \rmdefault}{pmnx}}
\DeclareOption{oldstyle}{\renewcommand*{%
 \rmdefault}{pmnj}}
\ExecuteOptions{oldstyle}
\ProcessOptions*
\endinput
```

5.3 Selecting font variants

Now we know how to select a given font family, which may feature a large number of font weights, widths, and variants within it. Once we have selected a font family, how can we direct TEX to select from our many alternatives within that font family, such as boldface, italic, small-caps, and so on? The answer is to change the current series and shape.

While the low-level TEX commands (\fontseries and \fontshape) directly change the current series and shape, LATEX's higher-level commands are more commonly used. Most LATEX users know that \textbf selects boldface; LATEX implements this internally by setting the font series to \bfdefault, which is in turn defined as bx. Similarly, \textit (italics) utilizes italics by setting the font shape to \itdefault, defined as it. And \textsc (small caps) sets the shape to \scdefault, defined as sc.

We can use similar techniques to add more selection commands for more features that are not part of the LATEX core set of commands. Philipp Lehman's font installation guide is an excellent tutorial for this task [9]; it carefully constructs and explains a style file of NFSS extensions, nfssext.sty. We now take a closer look at how to support alternate weights and how nfssext.sty supports small caps with italics and switching between old-style and lining figures.

5.3.1 Supporting alternate weights

By default, LATEX supports a bold (\textbf) weight command. Let's say we feel the default bold is a little too dark, and we'd like to use semibold-condensed instead. We can accomplish this with a single line in our sty file:

```
\renewcommand*{\bfdefault}{sbc}
```

And now we like semibold-condensed so much, we'd like to add it as a new command, \textsb.

```
\newcommand\sbdefault{sbc}
\DeclareRobustCommand\sbseries
    {\not@math@alphabet\sbseries\mathsb
     \fontseries\sbdefault\selectfont}
\DeclareTextFontCommand{\textsb}{\sbseries}
```

For simple features, declaring a default value, a RobustCommand, and a TextFontCommand suffice to make the feature available within TEX.

5.3.2 Supporting small caps with italics

Lehman points out that italics and small caps are both in the same "variant" category, so the built-in \textit and \textsc commands do not work harmoniously together. Barring changes to the core LaTeX font selection primitives, text set to both italic and small-caps would only preserve the innermost setting.

nfssext.sty remedies this problem by declaring an si shape, analogous to it and sc, and its associated selection commands:

```
\newcommand*{\sidefault}{si}
\DeclareRobustCommand{\sishape}{%
 \not@math@alphabet\sishape\relax
 \fontshape\sidefault\selectfont}
```

It then changes the italic and small-caps commands to check the current shape before setting the new shape. Only if the current shape is italic and the new shape is small-caps, or vice versa, does it set the new shape to si. (Recall that we assigned the si code to small-caps + italic variants in Section 4.3.)

THE *RESULT* is *properly* *NESTED* small-cap, italic text.

```
{\textsc{The \textit{result}} is \textit{properly
\textsc{nested}} small-cap, italic text.
```

5.3.3 Supporting old-style and lining figures

Section 5.2 showed how to choose old-style or lining figures by default. However, it may be useful to have one as the default and use the other via an explicit command. In nfssext.sty, the new commands \textos selects old-style figures and \textln lining figures. In this article, for instance, old-style is the default, so 1234\textln{1234} results in 12341234.

Because each style of figures is associated with a different font family, using an alternate figure style requires changing the family. nfssext.sty accomplishes this task as follows. Switching to lining figures for the font named abc first tries font family abcx then font family abc, using the TeX primitive \selectfont; switching to old-style figures switches to font family abcj. Fortunately this complexity is all hidden inside nfssext.sty.

6 Tools

For any OpenType font installation into TeX, the vital tool is otftotfm [8]. However, otftotfm only installs from a single OpenType font file with a single set of options, while users typically would like to install an entire family of OpenType font files with all available options. In addition, otftotfm does not address the problem of creating fd or sty files.

To address these issues, Marc Penninga wrote autoinst [13] and John Owens wrote otfinst [12], both of which wrap around otftotfm to install entire TeX font families. The two tools have far more similarities than differences and should suffice for most users; the authors' use of Perl (autoinst) or Python (otfinst) may make the difference for programmers familiar with one or the other.

Among the features supported by both tools are:

- A command-line interface that takes one or more OpenType font files as arguments;
- Installation of font families with the following features if present: roman and italic text; small-caps; lining, old-style, superior, and inferior figures; and swashes;
- nfssext.sty font selection commands;
- Support of optical sizes; and
- Auto-generation and installation of sty and fd files.

Some of the differences are that autoinst also supports numerators, denominators, upright swash, and titling text, and generates ornaments, while otfinst supports a scaling option at runtime. otfinst uses fontname naming, while autoinst is more verbose in its naming scheme. Finally, otfinst uses the metadata associated with each OpenType font to determine the font's characteristics, while autoinst extracts the characteristics from the font's filename.

6.1 Other tools

Geoffrey Washburn's otftofd [16] automates the process of creating fd files from OpenType fonts. Washburn indicates that it, like autoinst, is designed primarily for Adobe font conventions. otftofd is a shell script written in the Caml Shell and uses otftotfm.

The MinionPro TeX package [2] provides extensive TeX support files for Adobe Minion, including comprehensive options for figure types, encodings, ornaments, letterspaced small caps, and calligraphic, math, blackboard, and Greek fonts. The MinionPro distribution was built using otftotfm and thus contains all TeX support files without the need for the steps described in this article. MinionPro includes a package called fontaxes that extends (and partially replaces) NFSS's rigid classification of font attributes.

7 Conclusions

This article has described the steps necessary to use OpenType fonts in TeX: use otftotfm to install TeX font metric and encoding files; build a font description file for each font family; and build a style file for convenient font selection in a document.

OpenType handling in TeX is still far from ideal, however. Systems like XeTeX [7] use OpenType fonts natively with truly impressive results, and native OpenType support is slated for future versions of pdftex [4]. However, equally important are the other components of TeX that relate to font selection and invocation.

- NFSS is insufficient to elegantly describe the wide variety of available font attributes. The combination of weight and width into series is awkward, multiple variants may combine into a single shape, and features such as figure styles are not covered at all. The ideal font selection scheme not only includes all variants of a typeface but also allows simple, orthogonal selection of any set of typeface features, and transparent substitution when features are not present. New, flexible approaches such as fontspec in XeTeX [14] and MinionPro's fontaxes are encouraging steps toward such a scheme.
- Even within NFSS the codes are not standardized. Lehman's scheme appears to be widely used, however, which is encouraging.
- Even for simple features, font selection is wholly nonstandardized and non-orthogonal. Selection of alternate widths is not possible without low-level commands, the default italic and small caps commands do not work together because LaTeX's default handling of the "variant" category does not address multiple variants, and using non-standard but desirable selection commands such as \textln in shared files is discouraged because a default TeX installation does not support them.
- Finally, while many advanced OpenType features can be supported in TeX's font files, *invoking* those features with high-level commands is a much more troublesome task. For most features beyond the basic ones, TeX and LaTeX have no standardized support at the author level (or, in many cases, no support at all). Features like ornaments, contextual alternates, and discretionary ligatures would benefit from a discussion about how they can be invoked by the programmer in a standard way.

Acknowledgements Many thanks to Karl Berry for encouraging me to write this article and for his helpful suggestions along the way. Eddie Kohler's tools greatly ease the task of OpenType support in TeX, and I also thank Eddie for his prompt and thorough answers to my many questions about his tools. Karl Berry and Philipp Lehman were both quite helpful in understanding fontname and how it's used within the TeX world. The use of Philipp's nfssext.sty was vital in the development of otfinst. Thomas Phinney, Geraldine Wade, and Michael Duggan provided fonts for testing, and Thomas secured permission to use Adobe fonts for the article itself. Finally, thanks also to Nelson Beebe, Stephen Hartke, Oleg Katsitadze, Eddie Kohler, Marc Penninga, Will Robertson, and Michael Zedler for their thoughtful comments on the article during the review process.

References

[1] Karl Berry. Filenames for fonts. *TUGboat*, 11(4):517–520, November 1990. http://www.tug.org/fontname.

[2] Achim Blumensath, Andreas Bühmann, and Michael Zedler. MinionPro, September 2005. http://www.ctan.org/tex-archive/fonts/minionpro/.

[3] Maarten Gelderman. A short introduction to font characteristics. *TUGboat*, 20(2):96–104, June 1999.

[4] Taco Hoekwater. Opening up the type. *TUGboat*, 27(1):16–17, 2006.

[5] Alan Hoenig. *TeX Unbound*. Oxford University Press, New York, NY, 1998.

[6] Alan Jeffrey, Rowland McDonnell, and Lars Hellström. fontinst: Font installation software for TeX, December 2004. http://www.tug.org/applications/fontinst/.

[7] Jonathan Kew. The XeTeX typesetting system, 2006. http://scripts.sil.org/xetex.

[8] Eddie Kohler. LCDF type software, 2006. http://www.lcdf.org/type/.

[9] Philipp Lehman. The font installation guide, December 2004. http://www.ctan.org/tex-archive/info/Type1fonts/fontinstallationguide/.

[10] Adam T. Lindsay. OpenType installation basics for ConTeXt. *The PracTeX Journal*, 2005(2), April 2005.

[11] Frank Mittelbach and Michel Goossens. *The LaTeX Companion*. Addison-Wesley, Boston, MA, second edition, 2004.

[12] John Owens. otfinst, 2006. http://www.ctan.org/tex-archive/fonts/utilities/otfinst/.

[13] Marc Penninga. fontools, 2006. http://www.ctan.org/tex-archive/fonts/utilities/fontools/.

[14] Will Robertson. Advanced font features with XeTeX—the fontspec package. *TUGboat*, 26(3):215–223, 2005.

[15] LaTeX3 Project Team. LaTeX2ε font selection, June 1994. http://www.ctan.org/tex-archive/macros/latex/doc/fntguide.pdf.

[16] Geoffrey Washburn. otftofd, 2005. http://www.ctan.org/tex-archive/fonts/utilities/otftofd/.

◇ John D. Owens
Department of Electrical and Computer Engineering
University of California
One Shields Avenue
Davis, CA 95616 USA
jowens (at) ece dot ucdavis dot edu
http://www.ece.ucdavis.edu/~jowens/