The `VFinst` virtual font installer (version 1.00)

Alan Hoenig

November 30, 2007

If you've ever wanted to set up outline fonts for use by TEX or LATEX, `VFinst` may be of use to you. `VFinst` sets these fonts up for use by TEX and LATEX, and at the end of the `VFinst` process, you'll be able to typeset with your new fonts, confident that accents, ligatures, and the like will appear in your document with the usual TEX conventions. Here are the fonts you get at the end of the process.

1. For every upright font: you get the upright font, the small caps font, an oblique font, and an underline font.

2. For every italic font, you get the italic font, a small caps italic, an unslanted italic, and an underline font.

3. Additional fonts: You will also get titling (display) fonts, both upright and italic, and a swash italic font, if the appropriate raw fonts were present.

`VFinst` takes care to use expert fonts to build the small caps fonts, if these raw fonts were originally present. You also get test files for plain TEX and NFSS (LATEX), to both test the fonts and to provide examples of how to declare and use each of the fonts in a TEX document.

At the end of the `VFinst` process, you get all `.vf` and `.tfm` files for all fonts, which have been renamed for you according to the `fontname2.1` conventions. All oblique and italic fonts use the true italic angle for the font. The *dvips* map file `psfonts.map` will have been updated for you. All fonts will have been placed in the proper area on your hard disk. You may choose original TEX (`ot1`) or Cork (`t1`) encodings for your fonts, which can be installed according to the TDS (TEX Directory Structure) or 'traditional' standard. All raw fonts follow the `8r TeXBase1Encoding`.

All this and more is expanded upon in the remainder of this file. **Please read the file thoroughly.** The first section, a general introduction, is followed by four sections that a new user will need:

- '`VFinst` and what you'll need to run it',
- 'Installation and set up',
- 'Running `VFinst`', and

- 'Using your new fonts'.

The remainder can be read later (if at all).

If this software proves useful to you, you may also wish to investigate two other packages I wrote, both on CTAN. *MathInst* helps install commercial math fonts with your newly installed PostScript fonts so that equations and such come out looking fully compatible with your text. *MathKit* does the same thing, but it uses Metafont to create symbols from the Computer Modern fonts that match your text fonts. These packages can be found in the

```
fonts/utilities/mathinst
fonts/utilities/mathkit
```

areas of CTAN.

1 Introduction

One thing that *should* be easy to do with TEX is make commercial outline fonts transparently usable by TEX and LATEX. Even though a tool like `fontinst` does make it easy to do this, `fontinst` still requires inside knowledge not easily come by. The set of tools called `VFinst` is my attempt to make the task of installing families of PostScript fonts as simple as possible, but **only** for well-formed families of fonts. That is, I am concerned here only with *text* fonts, roman fonts appearing in medium and other weights, with italic variants, and optionally with small caps or expert fonts, and possibly a few additional matching fonts, like swash italic caps, display fonts, and so on thrown in. Special dingbat fonts, unusual families like Poetica, foreign language fonts, math fonts, and the like will not be reliably installed with this software. Neither will fonts that are narrow, extended, or other 'width' variants. (I have prepared two other tools—*MathInst* and *MathKit*—that help typeset documents so that the mathematics is compatible with the text fonts. Both these tools are available on CTAN; see above.)

Here's the basic idea. You place copies of all the fonts you want to install in a special work place. Then, you activate the `VFinst` scripts, run all the additional scripts that `VFinst` produces, and that's it! `VFinst` does these things for you:

- produces all `.vf` and `.tfm` files;
- renames the fonts according to version 2.1 of Karl Berry's `fontname` convention;
- automatically updates the `psfonts.map` file for *dvips*;
- generates two test files, for plain TEX and NFSS (LATEX), showing how to use your new fonts; and
- generates a script or batch file for placing all new fonts and other associated files in their proper places (It also generates a script for creating any needed new directories.)

and does these for either 'traditional' or TDS-compliant systems. (TDS stands for 'TEX Directory Structure'.) Once you've executed all these scripts or batch files, the fonts are ready for use.

Here are the fonts you get:

- for each upright font (roman and bold or semibold, usually):
  - the font itself,
  - matching small caps (mock or real),
  - a slanted (oblique) font, and
  - an underline font.
- for each italic font (roman and bold or semibold, usually):
  - the font itself,
  - matching small caps,
  - an unslanted italic, and
  - an underline font.
- display or titling fonts (both upright and italic), if they are present; and
- swash italic fonts, if they are present.

If you have included expert or small caps fonts among your fonts for installation, `VFinst` automatically takes them into account when building the virtual fonts. If you have included expert raw fonts, you actually get two font families at the end—both use 'expertised' fonts, but one family contains regular digits, and the other contains old style digits.

It's not well known, but the angle of inclination for an italic font varies quite a bit from font family to font family. The oblique and unslanted italic fonts that `VFinst` creates use the true italic angle for that font. You also get several underline fonts, font in which each character is underlined so it is easy to typeset lengthy underlined passages without worrying about line breaks.

(This paragraph is for reference purposes only; it mentions things that will very likely not be of interest to most people.) The virtual fonts created use raw fonts which follow the '8r' `TeXBase1Encoding` being promulgated by the international community of TEX developers. (This encoding provides easy access to all characters in the font. Pardon to all for the cryptic acronyms that abound.) The resulting virtual fonts themselves follow either the original TEX encoding (so-called `OT1` or `ot1`) or the newer Cork encoding (`T1` or `t1`).

Pierre MacKay's `makevf` script helped inspire `VFinst`. The remainder of this document describes the `VFinst` installation procedure, its use, and other important details in greater detail.

Although I hope this software is useful, it comes with no warranty whatsoever. I cannot accept responsibility for any harmful effects from this software.

1.1 Changes from prior versions

Separate DOS versions of `VFinst` are no longer supported.

Several bug fixes are particularly important. One odd but interesting bug prevented `VFinst` from working properly with New Century Schoolbook fonts. A second concerned the way *dvips*-map entries were generated for TDS systems. Both have been fixed.

Speaking of TDS systems, this version has been very thoroughly tested with such systems, a fact which has not been true in the past. Several major bugs were uncovered and fixed, and my apologies to all that they persisted for so long.

1.1.1 Lousy afm files

The major problem users may continue to have lies not with *this* software, but with the `afm` files that accompany the files. These files provide the metric and other information for a Type1 font analogously to TEX's `tfm` files. Apparently, little software takes advantage of these files, and digital foundries are at best sloppy when preparing them. `VFinst` relies on the `afm` files to learn things about the fonts it's processing, and sloppily prepared `afm`s lead to disaster. This version of `VFinst` attempts to second-guess an `afm` file as much as possible. A variable called `@sloppy_afms` contains a list of particularly sloppy `afm`. Currently, it refers to Adobe Garamond and Minion, Monotype Bulmer, Lucida Bright, and BitStream Chianti. You can add to it if you feel you've discovered another malefactor. (See the beginning of file `vfinst.lib` for this variable.) Another mechanism for correcting `afm` files lies in the file `vfinst.rc`; see below.

1.2 Scope of testing

This software was tested on a Linux box and under WindowsNT 4.0 using both TDS and traditional versions of TEX. `VFinst` was extensively tested on several popular font families, including the PostScript-resident fonts Times Roman, Palatino, and New Century Schoolbook, as well as Adobe Garamond, Adobe Caslon, Adobe Minion, Monotype Bulmer, and Monotype Baskerville, Lucida Bright, plus many other families of fonts that were lying around on my hard disk. They all installed fine for me.

1.3 Legal jargon and caveats

This software is distributed according to the terms of the GNU 'copyleft' agreement. In addition, please note that if you make any alterations to any of the files in this package, you must change the names of the altered files. You are allowed to freely use this material and to pass it along (in the unaltered state!) provided you will not be using it to make money; otherwise, you must get in touch with me to secure permission.

While the author hopes that it will prove useful, it is distributed with no guarantee or warranty whatsoever. Users should take GREAT CARE to carefully back up their data before using.

2 `VFinst` and what you'll need to run it

`VFinst` itself consists of Perl scripts and some supporting files.

2.1 Example output

In the subdirectory `example`, you'll find the `pfb` and `afm` files for the Roman typeface Utopia, placed in the public domain by URW. In addition, you'll find other files that are the result of the `vfinst` process.

2.2 What you'll need

Here's what you'll need to properly install your fonts using `VFinst`:

1. a version of TEX that understands virtual fonts (version 3 or better of TEX);
2. the `fontinst` package available from any CTAN archive or mirror in the area

   ```
   fonts/utilities/fontinst/inputs
   ```

   Installation of this package is easy—you simply take **all** of the files in this directory and place them in a directory on your system that TEX know how to read from. For traditional systems, this is apt to be a place like '`tex/inputs`' and on a TDS system, a place like

   ```
   texmf/tex/generic/fontinst
   ```

   is appropriate. (There are apparently several different versions on CTAN. You want version 1.5 or greater. Versions before that may not work.)
3. the *dvips* post-processor;
4. a copy of the file `8r.enc`, part of the `fontname` package (this package is available from any CTAN archive from the area `info/fontname`); and
5. a copy of Perl. Perl is a powerful language for text processing. I chose Perl because versions of Perl are freely available for every computer platform. You don't need to understand Perl—just get a copy of the executable file, place it on your computer's path, and (if necessary) make it executable. This material has been tested under Perl5 (the current version). Perl should be available from any good software archive and from many CDROM collections of software.

This material should run on any platform on which runs Perl and TEX.

3 Installation and set up

3.1 Installation

To install `VFinst`, create a new directory, possibly something like

```
/usr/local/vfinst
```

under Unix or

```
\tex\vfinst
```

under DOS. Copy the all the `VFinst` files (with the possible exeception of the examples) into that directory.

Now create a work directory below this new directory, with a full pathname like

```
/usr/local/vfinst/work
```

for Unix or

```
\tex\vfinst\work
```

under DOS. Switch to this lowest level `work` subdirectory, which we will call the `VFinst working` directory. If you are on a Unix system, make sure you have permission to write files in this working area.

Also, make sure that your implementation of TEX contains the *parent* directory and the *current* directory on its own search path.

There are some things that need setting in the file `vfinst.par,` which is one of the files in the `VFinst` directory. **Everyone** must set these up properly—`VFinst` won't work properly otherwise, and there are no default settings for these parameters. Take the time to doublecheck the settings for accuracy.

Since this is a Perl file, you must follow Perl syntax in the setup. This isn't difficult; see below for examples of setups. In particular, the Perl comment character is `#`, which can hide or expose a line to Perl as the '`%`' does in a TEX document.

3.2 Everybody must set these...

1. `$vfenc` is the encoding you'll want to be using. Although many in the TEX community are pushing an encoding called `t1,` I prefer to use the original TEX encoding called `ot1` because there is room in these virtual fonts for many additional characters. Your only choice for this parameter is `t1` or `ot1.` Users of versions of LATEX predating December 1996 might enter `T1` or `OT1` instead.

   If you use the lowercase `ot1` or `t1` designations, you'll need to *make sure* that corresponding files `ot1*.etx` and `t1*.etx` are in a TEX input directory. If such named files do not accompany your copy of `fontinst,` the simplest thing to do is make a copy of all the `OT1*.etx` and `T1*.etx` files, which are part of the `fontinst` package, and give them the appropriate names, eg `cp OT1.etx ot1.etx` (using Unix syntax).

In earlier versions of `VFinst`, a special variable contained the location of the file `8r.enc.` With this release, it is assumed visible to `dvips` without any path prefix. For traditional users, this may mean placing it in your `work` directory. For TDS users, this means nothing special, as `dvips` will be able to find it via the search mechanism that accompanies the TDS installation.

3.3 TDS systems

TeX knows it's a TDS installation if the variable `$vftexmf` is defined; it should have as values the root TeX/MF directory. One advantage of TDS is that `VFinst` can then figure out for itself where most things belong.

In a TDS system, `VFinst` will ask you later for certain additional information pertaining to your fonts. For each font family, `VFinst` needs a typeface name (like '`agaramon`', '`times`', etc.) and a supplier (`adobe`, `monotype`, and so on). `VFinst` tries to figure this out by itself, and may simply request verification from you. In stubborn cases, it will ask you for this information.

3.4 Traditional systems

On traditional systems, all `.afm`'s go in one place, all `.tfm`'s in another, and so on. Such installations need to define the following variables. Directories are not well-structured in these kinds of systems, so `VFinst` needs lots of information from you, and this additional information belongs in `vfinst.par`.

1. `$vfvf` is the place where virtual fonts are kept. If you have vf's on your system already, use this directory name. Otherwise, you'll need to check your *dvips* or driver documentation to see where those programs expect to find the vf's. It should be something like

   ```
   "/usr/lib/tex/fonts/vf"
   ```

   (Unix) or perhaps

   ```
   "\\emtex\\fonts\\vf"
   ```

   (DOS; the backslash must be doubled as shown, or Perl will misunderstand it).

2. `$vftfm` is the place where TeX looks for your `.tfm` files.

3. `$vfpsfonts` records the location of your outline font files, files typically with extensions `.pfa` or `.pfb.`

4. `$vfafm` is the directory containing all the `.afm` files for the outline font files. Unlike most software out there, TeX needs these files.

5. `$vfmapdir` is the place where you keep the file `psfonts.map` that *dvips* uses.

6. `$vfinputs` is the place where the `.fd` and other input files belong.

3.5 Sample setups

The first part of the `VFinst` setup for a traditional Unix system might look as follows. These declarations are part of the file `vfinst.par.`

On a Unix TDS system, these settings might have the following form.

```
$vftexmf    = "/usr/texmf";
$vfencoding = "/usr"; # where is the file 8r.enc?
$vfenc      = "ot1";
```

Here is a sample Unix setup for a traditional TEX system.

```
$vfmapdir   = "/usr/lib/tex/ps";
$vfinputs   = "/usr/lib/tex/inputs";
$vfencoding = "/usr";
$vfenc      = "ot1";
$vfvf       = "/usr/lib/tex/fonts/vf";
$vftfm      = "/usr/lib/tex/fonts/tfm";
$vfpsfonts  = "/psfonts";
$vfafm      = "/psfonts/afm";
```

On a DOS system, these instructions might take the form

```
$vfmapdir   = "\\tex\\ps";
$vfinputs   = "\\tex\\inputs";
$vfencoding = "$vfinputs";
$vfenc      = "t1";
$vfvf       = "\\tex\\fonts\\vf";
$vftfm      = "\\tex\\fonts\\tfm";
$vfpsfonts  = "\\psfonts";
$vfafm      = "\\psfonts\\afm";
```

(note the doubled backslashes).

3.6 Other parameters & considerations

The other 'wizards only' parameters should already have been set for you.

If your system understands the 'shebang' convention (whereby you type the name of script, and it knows to invoke Perl and run like a named program; the first line of the script must begin with '`#!`'), then all you'll need to do is adjust the initial line of files `1vfinst` and `2vfinst` to reflect the location of your Perl executable. On Unix systems, you'll also need to make these scripts executable via the `chmod +x` command.

If your version of TEX is a multitasking application, such as Tom Rokicki's excellent TEXView on NextStep, you'll need to make the application quit and restart it to make your new fonts visible to it.

4 Running `VFinst`

To use `VFinst`, simply copy all the fonts (outline file plus `.afm` file) to the `VFinst` `working` directory. `VFinst` won't care what filenames these have, nor how many font families are represented. If you have any expert, small caps, extension, or alternate fonts, include them too by all means. `VFinst` will use them if present to make true small caps fonts. **You MUST make sure that the outline font file and the matching `.afm` have identical filenames.**

Then, simply execute the following sequence of commands:

- `1vfinst`
- `2vfinst`

Under the 'shebang' notation, you need simply enter something like `../1vfinst`; otherwise, type something like `perl ../1vfinst`. These two scripts execute rapidly, although they pause to request information or verification from you. They will try to figure out the font supplier, the font family, and some TDS directory information, and it is this material that needs to be verified and/or provided.

Sometimes digital foundries are sloppy about naming their fonts consistently, and so `VFinst` may ask you to verify or enter a font family abbreviation for one family more than once. For example, Monotype Baskerville has been assigned to the family named `BaskervilleMT` while the matching expert font is in the family `BskvillExpMT`. (These are the names appearing in the `.afm` file.) A TEX author clearly needs them in the same font family, so verify family `mbv` for the first, and verify family `mbv` for the second, so that `VFinst` will know they are in the same family.

It often pays to briefly examine the files that `1vfinst` produces. There is a small file `duplicat.lst` which contains a list of fonts that `VFinst` thinks are duplicates. It will help later steps proceed faster if you eliminate any duplicates—and if you do, don't forget to remove both `.pfb` and `.afm` file. Entries in this file may also indicate a problem with the `.afm` file; see page 12 below for more on this.

The other output file is `fonts.lst,` a list of all fonts with their proposed new names, old names, and some information that will be of interest to later steps in the `VFinst` process. You should examine this list and eliminate any special-purpose fonts, such as special fraction or ornament fonts, from the installation process. It will also be helpful to eliminate all fonts with a width variant (such as Helvetica Narrow, etc.).

If you remove any fonts from your working directory, make sure to re-run `1vfinst.`

There are several additional steps. Execute the following scripts, which are outputs of the previous two steps:

- `mkdirs.bat`
- `newnames.bat`

to create any missing directories and to rename the fonts according to the `fontname2.1` conventions. These steps take very little time.

Then, execute the command

```
tex makefont
```

to create the virtual fonts; the file `makefont.tex` is the output of the original Perl scripts. This step may take a long while. Ignore warnings about missing glyphs.

Next, execute the script

- `maketfm.bat`

to convert your Ascii property list files into the binary `.tfm` and `.vf` files that TeX expects. Finally, execute the two scripts

- `putfonts.bat`
- `putvftv.bat`

These two scripts, which execute rapidly, are responsible for placing all files into places from which TeX can use them for typesetting. **The fonts cannot be said to be installed until these two steps are carried out.** I suppose that users on multi-user systems will have to request their system administrator perform these tasks.

One special step is necessary for TDS systems only! Whenever new fonts are added to a TDS system, it will be necessary to run a small utility which accompanies TeX. This tool may be named `texhash` or `configure` or `mktexlsr` something else entirely, and is necessary to place the names of the new fonts in database that TDS uses to locate files and fonts.

Now, your new fonts are ready for use. You can test your fonts by running either of the files `testpln.tex` or `testltx.tex` through `plain` TeX or through LaTeX. These files also show what commands to use to select your new fonts.

Once the installation has been completed according to the above steps, everything left in the work directory may be deleted.

4.1 Installing 'raw' fonts

`VFinst` is intended for fonts that require auxiliary constructs called *virtual fonts* for proper use by TeX. This does not include every font you might purchase. For example, dingbat fonts, special alphabets (phonetic fonts, foreign languages, APL) and so on don't require the virtual font mechanism. Installation to make these fonts usable by TeX is a much simpler process, but you have to do the work yourself. To illustrate the process, let's suppose you have files `foo.afm` and `foo.pfb` for installation. Follow these steps:

- You'll need program `afm2tfm`, part of the *dvips* suite. Execute the command

  ```
  afm2tfm foo
  ```

  This creates a file `foo.tfm` for use by TeX, and displays a line like

  ```
  foo AGaramond-Regular
  ```

  (or whatever) on your console..

- Place this displayed line in the file `psfonts.map`. If it's a non-resident font, you'll need to adjust this line as follows:

  ```
  foo AGaramond-Regular <foo.pfb
  ```

  to automatically download the font to the printer at print time.

- Place the `.afm` and `.pfb` font files where they belong on your system disk. Place `foo.tfm` where it belongs.

- To select this font in a `plain` document, add a line like

  ```
  \font\myfont = foo at 13pt
  ```

  to make `\myfont` the command that selects this font (at thirteen points). In LaTeX, you can use the undocumented command

  ```
  \newfont{\myfont}{foo at 13pt}
  ```

  to accomplish the same goal.

5 Using your new fonts

The files `testpln.tex` or `testltx.tex` can help you test and preview your new fonts. These files also can be consulted to see how these fonts were selected; this will be most useful for users of `plain` TeX.

LaTeX users will, of course, select their fonts by selecting the proper family, series, and shape. Note that in the presence of expert fonts, `VFinst` creates two families—one that is expertised but with standard numerals, and another that is expertised and uses old style figures. In the absence of expert fonts, `VFinst` installs a single family (and creates small caps by means of judicious scaling of the font majuscules. Thus, if the font family name `foo` has no expert fonts, `VFinst` installs the raw fonts under the family `foo`. In the presence of expert fonts, you will have two families—`foox` and `foo9`. Note too the existence of several new font shapes which may be available for use by NFSS:

- `si` provides italic small caps;
- `t` provides a titling or display font;
- `ti` provides a titling or display italic;
- `sw` provides a swash italic variant;
- `un` provides an underline upright font; and
- `ni` provides an underline italic font.

`VFinst` will only provide these shapes in case the original raw fonts that were present support these shapes.

6 Causes for failure and other problems

`VFinst` maintains an extensive log file called `vfinst.vlg` which contains useful information which may be of use in case of disaster or other problems. (Another log file, `maketfm.vlg`, records information about the making of the binary font files but is generally of less usefulness.)

You may find that when you install too many fonts at once, TEX won't have enough font memory to load all of them at once. You may have to compile the test documents in parts—comment out a portion, test the remainder, and so on.

Installing too many fonts may lead to another rare error involving older versions of *dvips*. After a while, the `psfonts.map` file becomes too lengthy and *dvips* expires while complaining of being 'out of string space'. The easiest fix is simply to comment out entries in `psfonts.map` that you aren't currently using. The comment character is TEX's '`%`' symbol. Even better: simply update to a current version of *dvips*.

`VFinst` warns you on the rare occasion when the `fontname` name of a font exceeds eight characters; an example is the swash italic display characters in Adobe Minion. This is only a problem for DOS users, who should probably remove these fonts from the `VFinst work` directory and re-run `VFinst`.

6.1 Fixing `.afm` files

Virtual fonts that are created but don't typeset as you expect may indicate a problem with the font `.afm` files rather than with `VFinst`. Font providers are often careless (at best!) with these files. Errors that I have encountered include misspelling the words 'bold' and 'italic' (now really!), improper specifying an x-height to be 0, and improperly naming the glyphs of the fonts. All these will effect the resulting fonts.

Entries in the file `duplicat.lst` may indicate other problems with the `.afm` file. While testing, I many times received warning that `VFinst` thought that one font was a duplicate of another when in fact one font was a regular weight and the second was bold. Closer inspection of the bold `.afm` revealed the problem—the ‘Weight’ entry the `.afm` file had erroneously been filled in as ‘`regular`’ even though it was a bold font. Basically, `.afm` problems involve correcting metric data in the body of the file or correcting non-metric data in the initial portion of the file. Both `fontinst` and `VFinst` provide means for correcting these errors. It is a bad, bad idea to make corrections to the `.afm` file itself.

Changes to data in the initial portion of an `.afm` involve creating a file called `vfinst.rc` each line of which records three items—the font name of the deficient font, the name of the characteristic to change, and the corrected value of that characteristic. Each of these fields should be separated by one or more spaces. The font name is the long font name that follows the keyword ‘FontName’ in the first part of the `.afm` file (within the the first dozen lines from the top of the file).

The `.rc` file that comes with the current distribution looks like this:

```
CentaurSwashMT            FullName       Centaur Swash Italic MT
Courier                   Weight         Roman
Courier-Oblique           Weight         Roman
Helvetica                 Weight         Roman
JansonExpertMT-BoldItalic ItalicAngle    -14
NewCenturySchlbk-Italic   Weight         Roman
Palatino-Italic           Weight         Roman
AGaramond-Bold            Weight         Bold
AGaramond-Semibold        Weight         Semibold
AGaramond-BoldItalic      Weight         Bold
AGaramond-SemiboldItalic  Weight         Semibold
AGaramondExp-Bold         Weight         Bold
AGaramondExp-Semibold     Weight         Semibold
AGarExpSemIta             Weight         Semibold
AGarExpSemIta             FullName Adobe Garamond Expert Semibold Italic
AGaramondExp-BoldItalic   Weight         Bold
AGaramond-BoldItalic      ItalicAngle    -18.5
AGaramondExp-BoldItalic   ItalicAngle    -18.5
AGarExpSemIta             ItalicAngle    -18.5
AGaramond-SemiboldItalic  ItalicAngle    -18.5
AGaramond-Italic          ItalicAngle    -18.5
AGaramondExp-Italic       ItalicAngle    -18.5
AGaramondAlt-Italic       ItalicAngle    -18.5
NewsGothic-Oblique        Weight         Medium
Garamond-BookItalic       Weight         Book
NewsGothic-BoldOblique    Weight         bold
```

This file indicates, for example, that `VFinst` is to regard the weight of the font `Courier` as Roman, the italic angle of `JansonExpertMT-BoldItalic` as $-14°$, and the full name of `CentaurSwashMT` as that given here. (The original font didn’t include the word ‘italic’ in the font name, so `VFinst` assumed the swash applied to upright fonts.)

I’d be grateful to anyone who could send me additional errors of this nature they encounter in using fonts, so I can encounter them in future revisions of this file.

7 Other operating systems

I designed these scripts to be easy to adapt to other operating systems. That, in fact, motivated my choice of Perl. All of the testing and development work was done in Unix, however.

Here are some things that need attention in going to another operating system. Change the three environment variables marked for 'wizards only'.

1. `$vfsep` is the directory separator in paths; for ex, `/` in Unix, `\` in DOS.
2. `$vfcopy` is the copy command (eg, `cp` and `copy` in Unix and DOS. respectively)
3. `$vfdel` is the delete command (for erasing files); it is `rm` in Unix and `del` in DOS.
4. `$mv` is the move command—the command that will move one file onto another even if the destination file already exists. In Unix, this command is '`mv`'. On DOS, it is '`call domove`' where `domove.bat` is a batch file with lines

   ```
   copy %1 %2
   del %1
   ```

   (The file `domove.bat` is part of the DOS distribution. You must remember to copy it to the `VFinst` working directory or to a place on your system's search path.)
5. `$ren` is the command for renaming a file.
6. `$rem` is the string that introduces comments in system ASCII files. It is '`#`' and '`rem`' in Unix and DOS.

Other places in the Perl files where it seemed to me would require system dependent changes are marked by the phrase 'system dependent' in comments in the code.

8 Additional info; authorial communications; acknowledgments

I'm truly grateful for the comments, suggestions, and trapped bugs reported by folks who took time to wrestle with this software. I especially want to acknowledge Andrew Chang, Burger Christian, Daniel Courjon, Colin Marquardt, David Ness (especially!), Seth Padowitz, Arend van Roggen, Wayne Sullivan, Matt Swift, Richard Walker, and Michael Yates.

For some additional information, please consult the book *TEX Unbound: LATEX and TEX Strategies for Fonts, Graphics, and More* by Alan Hoenig (New York and Oxford: Oxford University Press, 1998).

I'd love to hear from anyone with comments, criticisms, suggestions for improvements, and (of course) bugs that need fixing.

Alan Hoenig
`ajhjj@cunyvm.cuny.edu`
Department of Mathematics
John Jay College

445 West 59th Street
New York, NY 10019 USA

24 August 1998