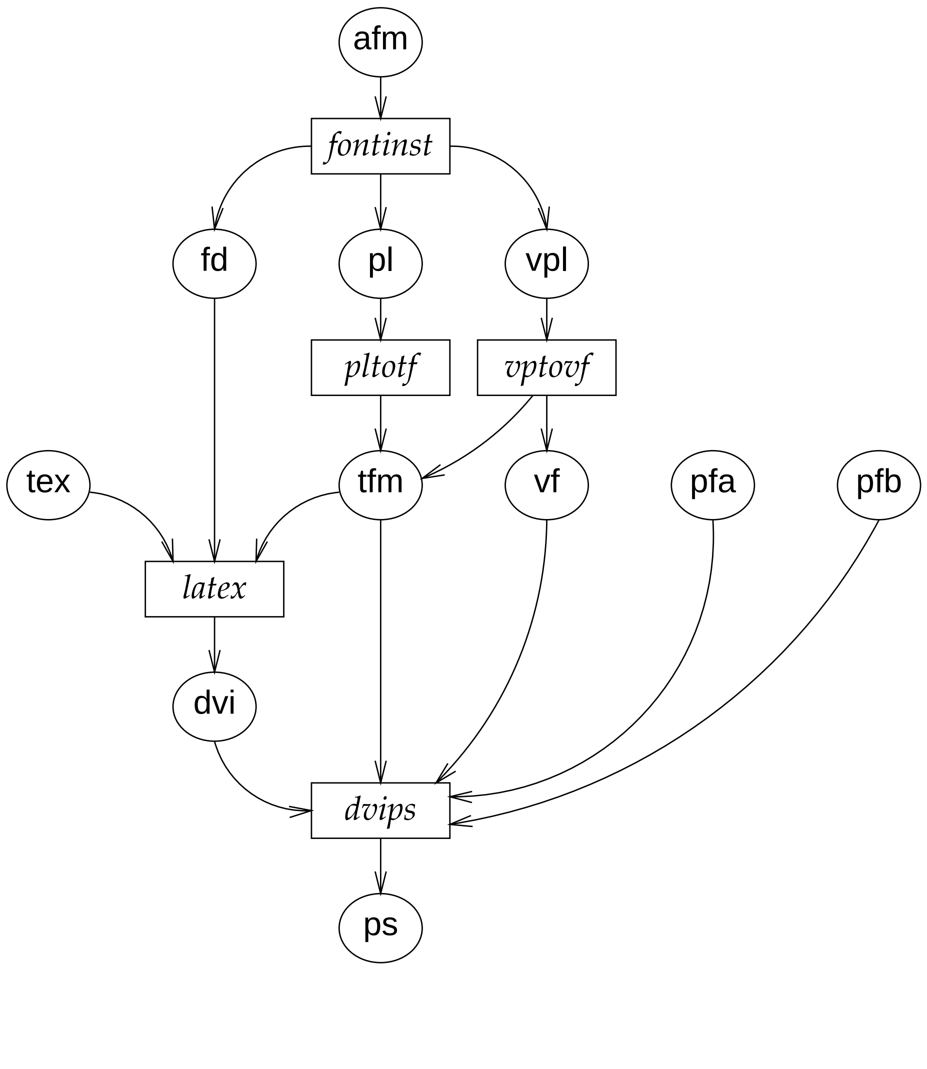
fontinst

Font installation software for TEX



Alan Jeffrey and Rowland McDonnell

fontinst v1.8 · 30 June 1998

This is the tutorial part of the 1998 FONTINST manual, chiefly written by Rowland McDonnell and conveying his then-view on related subjects. Although most of it is still correct, the passing of time has rendered it somewhat antiquated, and the average current (2004) user new to FONTINST would probably find Philipp Lehman's *The Font Installation Guide* (`http://www.ctan.org/tex-archive/info/Type1fonts/fontinstallationguide/`) more relevant.

Contents

1 Introduction

The FONTINST package is a set of TEX macros written to create virtual fonts for use with TEX. Its main use is creating the files needed so you can use PostScript Type 1 fonts with LATEX.

FONTINST needs information about the fonts it works with. This information needs to be supplied in an Adobe Font Metric (`afm`) or TEX Property List (`pl`) file. `pl` files can be created from `tfm` files using TFTOPL, a program normally included with a TEX system.

The job that FONTINST does is complicated, but it can be used for many tasks by people who are not TEX font wizards. Having said that, you do need to understand at least the basics of LATEX 2$_\varepsilon$'s font selection mechanism, which is documented in `fntguide.tex`, part of the standard LATEX distribution. `ftp://ftp.tex.ac.uk/tex-archive/macros/latex/base/fntguide.tex` will fetch a copy if you don't have one to hand.

To get the most benefit out of FONTINST, it's important to understand and use Karl Berry's 'Fontname' naming scheme. The definitive version of this is available from your nearest CTAN server. The following URL will fetch all the files needed compressed into a single ZIP archive: `ftp://ftp.tex.ac.uk/tex-archive/info/fontname.zip`. I suggest that you print out the Fontname documentation and have it handy when you're learning about FONTINST.

The FONTINST package:

- Is written in TEX, for maximum portability (at the cost of speed).
- Supports the OT1 (Computer Modern) and T1 (Cork) encodings.
- Allows fonts to be generated with arbitrary 'fake' characters; for example the 'ij' character can be faked if necessary by putting an 'i' next to a 'j'.
- Allows caps and small caps fonts with letter spacing and kerning.
- Allows kerning to be shared between characters, for example 'ij' can be kerned on the left as if it were an 'i' and on the right as if it were a 'j'. This is useful, since many PostScript fonts only include kerning information for characters without diacriticals.
- Allows the generation of math fonts with `nextlarger`, `varchar`, and arbitrary font dimensions.
- Allows more than one PostScript font to contribute to a TEX font, for example the 'ffi' ligatures for a font can be taken from the Expert encoding, if you have it.
- Can automatically generate a `fd` file for use with LATEX 2$_\varepsilon$.
- Can be customized by the user to deal with arbitrary font encodings.

Fontinst has been a stable piece of software since mid-1994. All further updates will be upwardly compatible with the interface described in this document.

1.1 What does fontinst do?

FONTINST is a tool written in TEX that can create the various extra files needed so you can use PostScript fonts with LATEX and TEX. It can read in `afm` files, and produces the necessary `vpl`, `pl`, and `fd` files to use the fonts (the human-readable `vpl` and `pl` files produced by FONTINST are turned into the machine-readable `vf` and `tfm` forms by VPTOVF and PLTOTF). It does not help you configure your DVI driver.

There also exists a `perl` front-end to FONTINST, intended specifically for use with a Unix TEX system, which takes care of routine tasks such as running VPTOVF and PLTOTF on the generated files after FONTINST has finished it's job. It also generates a font map file for use with DVIPS.

FONTINST's main job is creating `vf` files (virtual fonts). Not all TEX systems can use them. As far as I know, all current (1998) free and shareware TEX systems can; virtual fonts have been in widespread use with TEX since 1990. If you have a TeX system that can't use virtual fonts, FONTINST is most likely useless to you.

There are some nice things about having a tool written in TEX to do this: it's completely portable and you can modify its behaviour using TEX commands. The only real problem is that it's relatively slow: you can expect a typical FONTINST run to take something like 10–20 minutes on, say, a 40 MHz 80486SX PC or a 25 MHz 68LC040 Macintosh.

FONTINST can do its work on any font for which you have a corresponding `afm` or `tfm` metric file, so it's not limited to working with PostScript fonts; I have used it to produce the files I needed to use TrueType fonts with LATEX. Whether or not you can do this depends on whether or not you have suitable metric files and whether or not your TEX system can use TrueType fonts. In particular, the pdfTEX program supports TrueType fonts and includes a utility `ttf2afm` to generate `afm` files from `ttf` fonts.

Some people have used FONTINST to produce 'special effects' with normal TEX fonts. One example is the ECO set of fonts (available from CTAN: `ftp://ftp.tex.ac.uk/tex-archive/fonts/eco/`). These fonts are the same as the standard EC (European Modern) fonts, but with normal numerals replaced with old style numerals – 12345 rather than 12345 – everywhere except in maths mode.

1.2 Installation

To install FONTINST, put the contents of the `inputs/tex`, `inputs/etx`, `inputs/mtx` and `examples` directories into a directory read by TEX, for example `TEXMF/tex/generic/fontinst`.

When you use FONTINST, you need to make sure that the `afm` and `pl` files it will work on are in a directory searched by TEX.

If you are using `web2c` TEX on a Unix system with the TEX directory structure (TDS), you might put all the `afm` files in subdirectories of `TEXMF/fonts/afm/*`. And then say:

```
setenv TEXINPUTS $TEXMF/fonts/afm//::
```

Note that `pl` files are not normally kept in TEX installations, so if you want to use MF fonts with FONTINST you have to generate the corresponding `pl` files from `tfm` files and put them in your working directory before running FONTINST.

You could adopt a similar strategy with other TEX systems: create directories for the required files and then change the relevant parameter (`input_folders` in the default configuration file with OzTEX, for example).

The approach I use is this: I write a file containing commands for FONTINST to process, and put the `afm` and `pl` files needed in the same directory as that file. When FONTINST has finished working, I delete the `afm` and `pl` files because they are not needed and waste space on my hard disc drive. Some application programs on some computers need `afm` files, so it's not always a good idea to remove them completely.

1.3 Why do we need fontinst?

TEX refers to characters by number when it's typesetting. When you use a command like `\i`, TEX puts a number (16 if you're using OT1, 25 if you're using T1) into the `dvi` file. If you're using a font designed for use with TEX, this number will correspond to the character 'ı'. Assuming OT1 encoding for the moment, when you come to print out your `dvi` file, the DVI driver will see the number 16 in the `dvi` file, and select the character that sits in position 16 of the corresponding printer font file (a `pk` file in the case of normal TEX fonts). Unless something has gone wrong, that will result in the character 'ı' being placed on the page.

It's useful to think of these numbers and the actual characters corresponding to each number as sets called 'encodings'. A particular set of characters are assigned particular numbers. An example of an encoding is shown in table 1.

TEX began life using 7-bit fonts. This means the original TEX fonts used the numbers 0–127 to represent characters: 128 characters per font. TEX can now use 8-bit fonts: 256 numbers from 0–255, but even so, most typesetting with TEX still uses the original 7-bit encoding, now called 'OT1' (Old TEX 1 encoding). This has a correspondance between numbers and characters shown in table 1. The numbers used in that table are hexadecimal and octal because it makes for a neat table and anyway I stole the code to generate it from Donald Knuth and that's how he did it.

Returning to the example above, if you've a number 16 in your `dvi` file (expecting 'ı', a dotless i), but rather than printing with an OT1 encoded font, you print using a non re-encoded PostScript font in Adobe standard encoding, you'll get a blank, because the Adobe standard encoding has nothing in that character position.

There are several ways round this problem; I'll consider two cases here. If you are using LATEX you can tell it about a new encoding and re-define the commands that produce characters like 'ı' that live in different positions in

	´0	*´1*	*´2*	*´3*	*´4*	*´5*	*´6*	*´7*	
´00x	Γ	Δ	Θ	Λ	Ξ	Π	Σ	Υ	`"0x`
´01x	Φ	Ψ	Ω	ﬀ	ﬁ	ﬂ	ﬃ	ﬄ	
´02x	ı	ȷ	`	´	ˇ	˘	¯	˚	`"1x`
´03x	¸	ß	æ	œ	ø	Æ	Œ	Ø	
´04x	-	!	”	#	$	%	&	’	`"2x`
´05x	(	)	*	+	,	-	.	/	
´06x	0	1	2	3	4	5	6	7	`"3x`
´07x	8	9	:	;	¡	=	¿	?	
´10x	@	A	B	C	D	E	F	G	`"4x`
´11x	H	I	J	K	L	M	N	O	
´12x	P	Q	R	S	T	U	V	W	`"5x`
´13x	X	Y	Z	[	“	]	ˆ	˙	
´14x	‘	a	b	c	d	e	f	g	`"6x`
´15x	h	i	j	k	l	m	n	o	
´16x	p	q	r	s	t	u	v	w	`"7x`
´17x	x	y	z	–	—	˝	˜	¨	
	`"8`	`"9`	`"A`	`"B`	`"C`	`"D`	`"E`	`"F`	

Table 1: The OT1 font encoding

different encodings, or you can use a tool to re-encode the font so that it has the expected characters in the appropriate positions.

Re-encoding is the approach FONTINST uses: it can produce files to map the characters in the new font to one of TEX's existing encodings; this works with formats other than LATEX.

The first approach is used to define the standard encodings that LATEX uses. See, for example, the file `ot1enc.def` that comes with the current LATEX distribution, which defines the a few commands that refer to characters which aren't in the positions TEX would otherwise assume. This works only with modern versions of LATEX.

The second approach is used to allow you to use fonts in other encodings with any dialect of TEX. It has the some advantages over the first method: it works with any TEX format; and it improves portability, because you can typeset a document using a standard TEX encoding, sure that the same document will print correctly on a different kind of computer using a font with a different encoding. For example, you might say:

```
\usepackage{times}
```

in the preamble of your document. On my computer, that means my DVI driver will use a Macintosh encoded TrueType version of Times. On your computer, it might mean the dvi driver will use a Unicode encoded PostScript version of Times-Roman. The results will be identical in either case, without needing to modify the document.

1.4 How do you use fontinst?

FONTINST works on `afm` files named (more-or-less) according to Karl Berry's font naming scheme (see `ftp://ftp.tex.ac.uk/tex-archive/info/fontname` at CTAN). Let's say you want to use the Adobe Times fonts. You can get the metric files for this font from CTAN:

Location of file at CTAN	*Rename to*
`fonts/psfonts/adobeafm/base35/tib_____.afm`	`ptmb8a.afm`
`fonts/psfonts/adobeafm/base35/tibi____.afm`	`ptmbi8a.afm`
`fonts/psfonts/adobeafm/base35/tii_____.afm`	`ptmri8a.afm`
`fonts/psfonts/adobeafm/base35/tir_____.afm`	`ptmr8a.afm`

The new name is the name you should give the `tfm` files so that FONTINST understands what each file contains. The initial '`p`' means 'Adobe'; '`tm`' means 'Times'; '`b`' bold, '`r`' roman, '`i`' italic; and '`8a`' means 'Adobe standard encoding'.

The simplest use of FONTINST is to put the four `afm` files in the same directory as `fontinst.sty` and run TEX on `fontinst.sty`. At the `*` prompt type:

```
*\latinfamily{ptm}{} \bye
```

Some time later (about 17 minutes on my rather old computer), FONTINST will have finished, having created:

- Two `pl` files for each `afm` file
- One `vpl` file for each TEX font
- One `fd` file for each family

The `pl` files come in pairs: for example, `ptmb8a.pl` and `ptmb8r.pl`. The `8a` version has the same encoding as the original font; the `8r` version is re-encoded to `TeXBase1` (`8r`) encoding, and is the font that is the base on which the T1 and OT1 encoded versions are based on. The raw `8a` (Adobe standard) encoded font is not normally used.

These can be converted to TEX fonts using PLTOTF or VPTOVF. If you have OzTEX, launch OzMF, select PLTOTF (or VPTOVF) from the Tools menu, and say 'Do all files'.

If you use the `bash` shell on a Unix system, you can process all files using these one-liners at the `$` prompt:

```
$ for f in in *.pl;  do pltotf $f; done
$ for f in in *.vpl; do vptovf $f; done
```

(This assumes that PLTOTF and VPTOVF can deduce the file names of the corresponding `tfm` and `vf` files automatically.)

You should then:

- Move the `tfm` files to your TEX fonts directory (e.g. `TEXMFLOCAL/fonts/tfm/*`).
- Move the `vf` files to your virtual fonts directory (e.g. `TEXMFLOCAL/fonts/vf/*`).
- Move the `fd` files to your TEX inputs directory (e.g. `TEXMFLOCAL/tex/latex/psfonts/*`).

If your TEX installation is organized using the TEX directory structure (TDS), it is customary to subdivide the `tfm` and `vf` files into subdirectories by supplier and typeface name.

The `pl`, `vpl`, and `mtx` files are debris that can now be deleted. `mtx` files are font metric files FONTINST creates for its own use from `afm` and `pl` files. They're just more convenient for TEX to read than other forms – think of them as FONTINST readable `afm` and `pl` files.

By now, you have all the files in place to produce a `dvi` file using the new fonts. You can make Adobe Times the default roman font in your document by putting this in your preamble:

```
\renewcommand{\rmdefault}{ptm}
```

TEX will now happily produce a perfectly good `dvi` file including the new font, which your DVI driver will choke on because you've not yet told it about the new fonts. Exactly how you do this depends on the dvi driver, but they all need the same information: the name of a TEX font; the printer font name it corresponds to; some information to handle an re-encoding needed; and (in the case of a PostScript driver) perhaps an instruction to download the font to the printer. You don't need to download the Times font to a PostScript printer, because Times is built in to every PostScript printer.

If you use DVIPS, these lines added to your `psfonts.map` file will do the job:

```
ptmr8r   Times-Roman       "TeXBase1Encoding ReEncodeFont"  <8r.enc
ptmri8r  Times-Italic      "TeXBase1Encoding ReEncodeFont"  <8r.enc
ptmb8r   Times-Bold        "TeXBase1Encoding ReEncodeFont"  <8r.enc
ptmbi8r  Times-BoldItalic  "TeXBase1Encoding ReEncodeFont"  <8r.enc
ptmro8r  Times-Roman       "0.167 SlantFont TeXBase1Encoding ReEncodeFont"  <8r.enc
ptmbo8r  Times-Bold        "0.167 SlantFont TeXBase1Encoding ReEncodeFont"  <8r.enc
```

And now you can print a `dvi` file containing the new fonts. If you really do want to use the 'raw' `8a` encoded fonts for some reason, you need to add these lines to your `psfonts.map` file:

```
ptmr8a   Times-Roman
ptmri8a  Times-Italic
```

```
ptmb8a   Times-Bold
ptmbi8a  Times-BoldItalic
ptmro8a  Times-Roman       "0.167 SlantFont"
ptmbo8a  Times-Bold        "0.167 SlantFont"
```

Assuming that you're using the `fd` file that FONTINST has produced, and that you've asked for the Adobe Times family (`ptm`) in medium series (`m`) and upright shape (`n` for normal) using the NFSS font selection commands:

```
\renewcommand{\rmdefault}{ptm}
\rmfamily \mdseries \upshape
```

Assuming further that you are using the `fd` file `t1ptm.fd` produced by FONTINST, the TEX font (`tfm` file) `ptmr8t.tfm` will be selected by the above commands, as you can see from the relevant line in `t1ptm.fd`:

```
\DeclareFontShape{T1}{ptm}{m}{n}{<-> ptmr8t}{}
```

This is what happens:

- TEX typesets your document using the font metric file `ptmr8t.tfm`; this is the font that is put in the `dvi` file.
- DVIPS looks at the `dvi` file, and sees a reference to the font `ptmr8t`.
- DVIPS searches for a `vf` file corresponding to `ptmr8t`; when it finds `ptmr8t.vf`, it knows it has a virtual font on its hands.
- DVIPS follows the instructions in the `vf` file, which map characters in `ptmr8t.tfm` to characters in the font `ptmr8r.tfm`. That is, when it sees a number 25 in the `dvi` file (dotless i – 'ı' – in T1 encoding), it replaces it with a number 17, which is a dotless i in `8r` encoding.
- Then DVIPS looks up the name of each number according to the scheme given in the file `8r.enc`, and replaces each number with the name of the character, in this case, number 17 is listed as 'dotlessi'.
- And finally, DVIPS tells the printer to print the named character.

Not all DVI drivers can manage re-encoding as well as DVIPS can. For example, OzTEX's built-in non-PostScript `dvi` driver can only work with numbers, so if I'm using a PostScript font, I can't print characters (such as Eth) that don't have a number in Macintosh text encoding unless I use DVIPS and print on a PostScript printer. DVIPS works with character names, so it's not subject to this restriction. In the example above, OzTEX would replace the number 17 for 'dotlessi' in `8r` encoding with a number 245 for 'dotlessi' in Macintosh text encoding.

The details of the LATEX 2ε font selection scheme are described in *LATEX 2ε font selection* (distributed with LATEX 2ε as the file `fntguide.tex`) and *The LATEX Companion* (Goossens, Mittelbach and Samarin, Addison-Wesley).

The files you need to use Times, Helvetica, Courier, and the rest of the 'standard' PostScript fonts are distributed as part of the PSNFSS bundle available from CTAN, so there's no need to create new files to use these fonts.

A more involved example of FONTINST use can be seen in the file `fontptcm.tex` which creates the files you need to use a combination of Times, Symbol, Zapf Chancery and Computer Modern as TEX math fonts.

2 Installing your own font family

The FONTINST package has a command `\latinfamily` meant to do most of the work to install a 'normal' set (family) of roman text fonts from Adobe. Assuming you have a set of `afm` files to match the fonts you wish to use, the first step is to rename the `afm` files according to the Fontname naming scheme.

A 'normal' set of text fonts usually includes the basic upright roman version, bold, italic, and bold italic. Sometimes there will also be small caps versions, perhaps some 'expert' fonts, and maybe some other weights such as light, medium, semi bold, black or ultra bold.

The most important point to note is this: no matter what sort of computer you're using and no matter what font encoding it uses normally, `afm` files for text fonts are almost always in `8a` encoding (Adobe standard encoding), so the `afm` files when renamed normally end in `8a`.

A typical set of four `afm` files re-named for use with FONTINST is this:

```
ptmr8r.afm    Times-Roman
ptmri8r.afm   Times-Italic
ptmb8r.afm    Times-Bold
ptmbi8r.afm   Times-BoldItalic
```

Not all `afm` files use `8a` encoding. If you open an `afm` file using a text editor, you'll see a line looking like this somewhere near the top:

```
EncodingScheme AdobeStandardEncoding
```

and if you see exactly that, the `afm` file should end with `8a`. If you see something like this:

```
EncodingScheme FontSpecific
```

have a look at the name of the font in the `afm` file. If you see something like this:

```
FontName AGaramondExp-Regular
FullName Adobe Garamond Regular Expert
```

you have an 'expert' encoded font on your hands, and the `afm` file should end with `8x` to indicate this to FONTINST. An `8x` encoded font contains extra glyphs like old style numerals, small capital letters, more ligatures, and so on.

The `\latinfamily` command is used like this:

`\latinfamily{`⟨*family*⟩`}{`⟨*commands*⟩`}`

This installs a Latin family of fonts.

For example, to install Adobe Times, you say:

```
\latinfamily{ptm}{}
```

The *commands* issued by LATEX each time a font from that family is loaded. This is most often used with typewriter fonts, to switch off hyphenation. For example, Adobe Courier can be installed with:

```
\latinfamily{pcr}{\hyphenchar\font=-1}
```

Once the installation is over (which may take some time) the fonts can be used in LATEX by selecting an appropriate `\fontfamily`, for example Adobe Times can be selected with:

```
\fontfamily{ptm}\selectfont
```

If the fourth letter of the family name is '`x`' then FONTINST will use expert fonts in creating the fonts. If the fourth letter is '`j`' (or for backward compatibility '`9`') then FONTINST will use expert fonts to create fonts with old style digits.

For example, to install Adobe Garamond using expert fonts, you say:

```
\latinfamily{padx}{}
```

To install Adobe Garamond using expert fonts with oldstyle digits, you say:

```
\latinfamily{padj}{}
```

When you have expert fonts, and you've told FONTINST to use them, it will carry on as normal, but the resulting font family will have the name '`padx`' or '`padj`', and it will use expert glyphs whenever possible, so you'll have a real (rather than faked) small caps font, real (rather than faked) 'ffl' ligatures, and so on.

Before using these commands, you will need to make sure that you have the Adobe Font Metric (`afm`) files for the fonts, and that they have appropriate names. The FONTINST package uses the LATEX convention for naming fonts, and uses a *font family* name which consists of:

- a *supplier* (or foundry), such as '`p`' for Adobe.
- a *typeface*, such as '`ad`' for Adobe Garamond.
- up to two *variants*, such as '`j`' or '`x`' for 'old style digits' or 'expert'.

`b`	Bitstream
`f`	'free' (public domain)
`h`	Bigelow & Holmes
`i`	ITC
`l`	Linotype
`m`	Monotype
`p`	Adobe (`p` for PostScript)
`r`	'raw' (obsolete)
`u`	URW
`z`	bizarre

Table 2: A partial list of foundries

`a`	alternate
`d`	display, titling
`f`	fraktur, handtooled
`j`	oldstyle digits
`n`	informal, casual
`p`	ornaments
`s`	sans serif
`t`	typewriter
`w`	script, handwriting, swash
`x`	expert

Table 3: A partial list of variants

`c`	small caps
`i`	italic
`o`	oblique (i.e., slanted)
`u`	unslanted italic

Table 4: A partial list of shapes

`ac`	Adobe Caslon
`ad`	Adobe Garamond
`ag`	Avantgarde
`bb`	Bembo
`bd`	Bodoni
`bk`	Bookman
`bv`	Baskerville
`ca`	Caslon
`ch`	Charter
`cr`	Courier
`fr`	Frutiger
`fu`	Futura
`gl`	Galliard
`gm`	Garamond
`gs`	Gill Sans
`hv`	Helvetica
`mn`	Minion
`lc`	Lucida
`lh`	Lucida Bright
`ls`	Lucida Sans
`nb`	New Baskerville
`nc`	New Century Schoolbook
`op`	Optima
`pl`	Palatino
`sy`	Symbol
`tm`	Times
`ut`	Utopia
`zc`	Zapf Chancery
`zd`	Zapf Dingbats

Table 5: A partial list of faces

So the family name '`padj`' indicates Adobe Garamond with old style digits. Note that the variants '`j`' or '`x`' are interpreted by FONTINST itself and do not appear in external font names, whereas other variants are passed through as part of the font names. (This is needed for families which have a sans serif or typewriter variant.)

The *supplier* must be one letter, and the *typeface* must be two (this is an attempt to fit all filenames into MS-DOS format). Each variant is one letter. The full list of foundries, typefaces, shapes and variants is given in Karl Berry's '*Filenames for fonts*' (available by anonymous FTP from `ftp://ftp.tex.ac.uk/tex-archive/info/fontname`), but the more common ones are given in Tables 2–5.

The FONTINST package uses Karl Berry's naming scheme for `afm` files. The full naming scheme is rather more flexible than the subset used by FONTINST, which uses filenames consisting of:

- a *supplier*, such as '`p`' for Adobe.
- a *typeface*, such as '`hv`' for Helvetica.
- a *weight*, such as '`r`' for regular.

`b`	bold
`c`	black
`d`	demibold
`h`	heavy
`k`	book
`l`	light
`m`	medium
`r`	regular
`s`	semibold
`u`	ultra bold
`x`	extra bold

Table 6: A partial list of weights

`c`	condensed
`n`	narrow
`w`	wide
`x`	extended

Table 7: A partial list of widths

`8a`	Adobe Standard
`8x`	Adobe Expert
`8r`	TEX 8-bit 'raw' (`TeXBase1`)
`8y`	TEX 8-bit 'raw' (`TeXnANSI`)
`7t`	TEX 7-bit text (`OT1`)
`7m`	TEX 7-bit math italic (`OML`)
`7y`	TEX 7-bit math symbol (`OMS`)
`7v`	TEX 7-bit math extension (`OMX`)
`8t`	TEX 8-bit text (`T1`)
`8c`	TEX 8-bit text symbols (`TS1`)
`9t`	TEX 7-bit text with expert glyphs
`9o`	TEX 7-bit text with expert glyphs and old-style digits
`9e`	TEX 8-bit text with expert glyphs
`9d`	TEX 8-bit text with expert glyphs and old-style digits
`9c`	TEX 8-bit symbols with expert glyphs and old-style digits

Table 8: A partial list of encodings

- up to two *shapes* or *variants*, such as '`o`' for oblique.
- an *encoding*, such as '`7t`' for Knuth's 7-bit TEX encoding.
- an optional *width*, such as '`n`' for narrow.
- a *file extension*, such as '`.tfm`' for TEX Font Metric.

So the filename name '`phvro7tn.tfm`' indicates Adobe Helvetica regular oblique narrow, in the 7-bit TEX encoding.

The full list of shapes, encodings and weights is given in Karl Berry's '*Filenames for fonts*', but the more common ones are given in Tables 4–6.

For example, to install Adobe Garamond including the expert fonts, you would need to rename the `afm` files:

Adobe name	*ATM name*	*Fontinst name*
`AGaramond-Bold.afm`	`gdb_____.afm`	`padb8a.afm`
`AGaramond-BoldItalic.afm`	`gdbi____.afm`	`padbi8a.afm`
`AGaramond-Italic.afm`	`gdi_____.afm`	`padri8a.afm`
`AGaramond-Regular.afm`	`gdrg____.afm`	`padr8a.afm`
`AGaramond-Semibold.afm`	`gdsb____.afm`	`pads8a.afm`
`AGaramond-SemiboldItalic.afm`	`gdsbi___.afm`	`padsi8a.afm`
`AGaramondExp-Bold.afm`	`geb_____.afm`	`padb8x.afm`
`AGaramondExp-BoldItalic.afm`	`gebi____.afm`	`padbi8x.afm`
`AGaramondExp-Italic.afm`	`gei_____.afm`	`padri8x.afm`
`AGaramondExp-Regular.afm`	`gerg____.afm`	`padr8x.afm`
`AGaramondExp-Semibold.afm`	`gesb____.afm`	`pads8x.afm`
`AGaramondExp-SemiboldItalic.afm`	`gesbi___.afm`	`padsi8x.afm`
`AGaramond-RegularSC.afm`	`gdsc____.afm`	`padrc8a.afm`
`AGaramond-SemiboldSC.afm`	`gdsbs___.afm`	`padsc8a.afm`

You can then run TEX on the following document to install the Adobe Garamond family:

```
\input fontinst.sty
\latinfamily{padx}{}
\latinfamily{padj}{}
\bye
```

Not all font families can be installed using the `\latinfamily` command, nor does it always produce optimal results. The main interfaces to FONTINST are at a slightly lower level, where all font names appear explicitly in command arguments, and at that level it is possible to fine tune the font generation. The `\latinfamily` command is mainly a clever collection of macros which expand to a mostly fixed[1] sequence of lower level FONTINST commands.

Descriptions of the sub-`\latinfamily` commands can be found in the main fontinst manual.

3 More on the `\latinfamily` command

The `\latinfamily` command is essentially a short-cut to save you preparing a huge file with many different FONTINST commands in it.

It takes `afm` or `mtx` files as the source of font metric data to work with. Usually, you have a set of `afm` files. They must be named according to a subset of the Fontname naming scheme. To illustrate the process, here is an edited part of the console log from a use of `\latinfamily`:

```
\latinfamily{pad}{}
```

This log does not show FONTINST 'in action'; it's just to illustrate which fonts are looked for when you use the `\latinfamily` command.

```
INFO> to make LaTeX font shape <pad,m,n,> seek padr8r.mtx
INFO> to make LaTeX font shape <pad,m,sc,> seek padrc8r.mtx
INFO> to make LaTeX font shape <pad,m,sl,> seek padro8r.mtx
INFO> to make LaTeX font shape <pad,m,it,> seek padri8r.mtx
INFO> to make LaTeX font shape <pad,m,n,c> seek padr8rn.mtx
INFO> to make LaTeX font shape <pad,m,sc,c> seek padrc8rn.mtx
INFO> to make LaTeX font shape <pad,m,sl,c> seek padro8rn.mtx
INFO> to make LaTeX font shape <pad,m,it,c> seek padri8rn.mtx
```

The important point to notice is that FONTINST needs an `8r` encoded `mtx` file for each font when you are using the `latinfamily` command. If it can't find an `8r` encoded `mtx` file, it'll look for for an `8a` encoded `afm` file. It will automatically turn the file it finds into an `8r` encoded `mtx` file. So when FONTINST says '`seek padr8r.mtx`', it is in fact looking for `padr8r.mtx` and `padr8a.afm`. Whatever it finds, it will end up with `padr8r.mtx` to work on.

[1]Commands that would refer to files which are not present are skipped, and in some cases there is a "Plan B" when "Plan A" would have made use of such a nonexistent file, but that is about it.

The first line of the log shows that FONTINST is trying to create a `vpl` file for `pad/m/n`. That is, font family `pad` (Adobe Garamond), font series `m` (normal 'book' or 'regular' weight), and font shape `n` (normal upright).

If it finds what it's looking for, it will create the files:

```
padr7t.vpl
padr8t.vpl
padr8c.vpl
```

And add these lines to the given `fd` files:

```
OT1pad.fd:    \DeclareFontShape{OT1}{pad}{m}{n}{<-> padr7t}{}
 T1pad.fd:    \DeclareFontShape{T1} {pad}{m}{n}{<-> padr8t}{}
TS1pad.fd:    \DeclareFontShape{TS1}{pad}{m}{n}{<-> padr8c}{}
```

This means you will have three new fonts to use in LaTeX: the OT1, T1 and TS1 encoded versions of `pad/m/n`. You'll be able to select (say) `T1/pad/m/n` by saying:

```
\fontencoding{T1}\fontfamily{pad}\fontseries{m}\fontshape{n}\selectfont
```

This is the clumsiest way of selecting that particular font, but I've done it to illustrate exactly what's happening.

The next line:

```
INFO> to make LaTeX font shape <pad,m,sc,> seek padrc8r.mtx
```

shows that FONTINST is trying to install a small caps font. If you have a real small caps metric file named `padrc8r.mtx` (don't forget it'll look for an `8a` encoded `afm` file), FONTINST will go ahead and create the `vpl` file and `fd` file entry as expected.

But you don't normally have a real small caps font, so FONTINST will quite happily produce a fake small caps font. To do this, it looks for a suitable metric file by dropping the '`c`':

> 'Hmm... I can't find `padrc8r`, so I'll look for `padr8r`.'

And you will eventually have:

```
padrc7t.vpl
padrc8t.vpl
```

And add these lines to the given `fd` files:

```
OT1pad.fd:    \DeclareFontShape{OT1}{pad}{m}{sc}{<-> padrc7t}{}
 T1pad.fd:    \DeclareFontShape{T1} {pad}{m}{sc}{<-> padrc8t}{}
```

(Note that it won't install a TS1-encoded small caps font because TS1 is a text symbol font, which would look the same in the upright and small caps shape.)

The next log line shows FONTINST trying to create a `vpl` for the oblique version of Adobe Garamond:

```
INFO> to make LaTeX font shape <pad,m,sl,> seek padro8r.mtx
```

It's quite usual for an oblique version to be unavailable, but FONTINST has a way round this: it can fake an oblique font from the corresponding 'straight' version:

> 'Oh dear: I can't find `padro8r`, so I'll look for `padr8r` and use clever maths to fake a slanted version.'

This is not as straightforward as the small caps case. FONTINST only works out what the metrics ought to be if the entire font is slanted to the right. It's up to the DVI driver to actually print a slanted font. DVIPS can do this.

You will eventually have:

```
padro7t.vpl
padro8t.vpl
padro8c.vpl
```

And these lines added to the given `fd` files:

```
OT1pad.fd:    \DeclareFontShape{OT1}{pad}{m}{sl}{<-> padro7t}{}
 T1pad.fd:    \DeclareFontShape{T1} {pad}{m}{sl}{<-> padro8t}{}
TS1pad.fd:    \DeclareFontShape{TS1}{pad}{m}{sl}{<-> padro8c}{}
```

The next line is straightforward:

```
INFO> to make LaTeX font shape <pad,m,it,> seek padri8r.mtx
```

If FONTINST can't find a suitable metrics file (`padri8r.mtx` or `padri8a.afm`), it carries on without doing anything. If it does find a suitable metrics file, it churns away until you will eventually have:

```
padri7t.vpl
padri8t.vpl
padri8c.vpl
```

And these lines added to the given `fd` files:

```
OT1pad.fd:    \DeclareFontShape{OT1}{pad}{m}{it}{<-> padri7t}{}
 T1pad.fd:    \DeclareFontShape{T1} {pad}{m}{it}{<-> padri8t}{}
TS1pad.fd:    \DeclareFontShape{TS1}{pad}{m}{it}{<-> padri8c}{}
```

The next line is a bit different. FONTINST is now trying to create a `vpl` file for a *condensed* font:

```
INFO> to make LaTeX font shape <pad,m,n,c> seek padr8rn.mtx
```

If it finds a suitable metric file (Adobe Garamond, medium weight, normal upright shape, condensed), it will eventually produce:

```
padr7tn.vpl
padr8tn.vpl
padr8cn.vpl
```

And these lines added to the given `fd` files:

```
OT1pad.fd:    \DeclareFontShape{OT1}{pad}{mc}{n}{<-> padr7tn}{}
 T1pad.fd:    \DeclareFontShape{T1} {pad}{mc}{n}{<-> padr8tn}{}
TS1pad.fd:    \DeclareFontShape{TS1}{pad}{mc}{n}{<-> padr8cn}{}
```

There is no standard LATEX command like `\bfseries` to select the medium condensed (`mc`) series created here. If you want to use this font, you must do something like:

```
\fontfamily{pad}\fontseries{mc}\selectfont
```

If it doesn't find a suitable metric file for a narrow series, FONTINST will just skip over and continue, unless you specifically tell it to fake a narrow series.

And so the process continues: FONTINST attempts to create `vpl` files for condensed versions of all the font shapes met so far, and then goes on to:

```
INFO> to make LaTeX font shape <pad,b,n,> seek padb8r.mtx
```

And again, if it finds a suitable metric file (`padb8r.mtx` or `padb8a.afm`), it'll potter off and create the files:

```
padb7t.vpl
padb8t.vpl
padb8c.vpl
```

And these lines added to the given `fd` files:

```
OT1pad.fd:    \DeclareFontShape{OT1}{pad}{b}{n}{<-> padb7t}{}
 T1pad.fd:    \DeclareFontShape{T1} {pad}{b}{n}{<-> padb8t}{}
TS1pad.fd:    \DeclareFontShape{TS1}{pad}{b}{n}{<-> padb8c}{}
```

With this step done, fontinst will try to create `vpl` files for the small caps, slanted, and italic versions of `pad/b`; and then it'll try to create condensed versions of all those:

```
  INFO> to make LaTeX font shape <pad,b,n,> seek padb8r.mtx
  INFO> to make LaTeX font shape <pad,b,sc,> seek padbc8r.mtx
  INFO> to make LaTeX font shape <pad,b,sl,> seek padbo8r.mtx
  INFO> to make LaTeX font shape <pad,b,it,> seek padbi8r.mtx
  INFO> to make LaTeX font shape <pad,b,n,c> seek padb8rn.mtx
  INFO> to make LaTeX font shape <pad,b,sc,c> seek padbc8rn.mtx
  INFO> to make LaTeX font shape <pad,b,sl,c> seek padbo8rn.mtx
  INFO> to make LaTeX font shape <pad,b,it,c> seek padbi8rn.mtx
```

If it manages to find the files it needs to create the `vpl` files to use all those fonts with LaTeX, you'll end up with the following lines in the T1 `fd` file (I've ignored the OT1 `fd` file to save some space):

```
  \DeclareFontShape{T1} {pad}{b} {n} {<-> padb8t}{}
  \DeclareFontShape{T1} {pad}{b} {sc}{<-> padbc8t}{}
  \DeclareFontShape{T1} {pad}{b} {sl}{<-> padbo8t}{}
  \DeclareFontShape{T1} {pad}{b} {it}{<-> padbi8t}{}
  \DeclareFontShape{T1} {pad}{bc}{n} {<-> padb8tn}{}
  \DeclareFontShape{T1} {pad}{bc}{sc}{<-> padbc8tn}{}
  \DeclareFontShape{T1} {pad}{bc}{sl}{<-> padbo8tn}{}
  \DeclareFontShape{T1} {pad}{bc}{it}{<-> pckbi8tn}{}
```

To translate into English: Adobe Garamond bold in 'normal', small caps, slanted, and italic versions, as well as condensed versions of all four.

Again, because there's no convenient way of selecting the condensed versions with existing LaTeX commands, you need to say something like:

```
  \fontfamily{pad}\fontseries{bc}\selectfont
```

to use the bold condensed (`bc`) versions of this font; you can of course use `\itshape`, `\scshape`, `\slshape`, and `upshape` to switch between the italic, small caps, slanted, and 'normal' versions of Adobe Garamond bold condensed once you've got `pad/bc` selected.

So far, you've seen `\latinfamily` look at two different weights and two different widths. For each weight, `\latinfamily` will try and install eight different fonts as you can see above. It will try and install the same eight different fonts for each of the following different weights:

LaTeX	*Fontname*	*description*
`ul`	`a`	ultra light
`el`	`i`	extra light
`l`	`l`	light
`m`	`k`, `r`	book, regular
`mb`	`m`	medium
`db`	`d`	demi bold
`sb`	`s`	semi bold
`b`	`b`	bold
`eb`	`c`, `h`, `x`	black, heavy, extra bold
`ub`	`u`	ultra bold

The LATEX column contains the label that will be used in the `\DeclareFontShape` command to specify the font series. The Fontname column contains the width specifier used to name the font metric file that FONTINST will look for in that case.

In other words, at some stage FONTINST will look for:

```
  INFO> to make LaTeX font shape <pad,sb,n,> seek pads8r.mtx
```

and if it finds a suitable metric file (`pads8r.mtx` or `pads8a.afm`), it will create:

```
  pads7t.vpl
  pads8t.vpl
  pads8c.vpl
```

and `fd` file entries like this:

```
  OT1pad.fd:    \DeclareFontShape{OT1}{pad}{sb}{n}{<-> pads7t}{}
   T1pad.fd:    \DeclareFontShape{T1} {pad}{sb}{n}{<-> pads8t}{}
  TS1pad.fd:    \DeclareFontShape{TS1}{pad}{sb}{n}{<-> pads8c}{}
```

and since `sb` is not a normal LATEX font series, you'll need to use something like:

```
  \fontfamily{pad}\fontseries{sb}\selectfont
```

to use this font.

The `basicex.tex` file in the `examples` directory of the main FONTINST distribution is an annotated command file which does roughly the same things as the `\latinfamily` command for the `pad` family of fonts.