

# 8 Fonts and layouts

This is the chapter that most users want first, because they come to structured documents from a wordprocessing environment where the *only* way to convey different types of information is to fiddle with the font and size drop-down menus.

As I hope you have seen, this is normally completely unnecessary in  $\text{\LaTeX}$ , which does most of the hard work for you automatically. However, there are occasions when you need to make manual typographic changes, and this chapter is about how to do them.

## 8.1 Changing layout

The design of the page can be a very subjective matter, and also rather a subtle one. Many organisations large and small pay considerable sums to designers to come up with page layouts to suit their purposes. Styles in page layouts change with the years, as do fashions in everything else, so what may have looked attractive in 1991 may look rather dated in 2011.

As with most aspects of typography, making the document readable involves making it consistent, so the reader is not interrupted or distracted too much by apparently random changes in margins, widths, or placement of objects. However, there are a number of different occasions where the layout usually *does* change, related to the frequency with which the format appears.

- The title page, the half-title, copyright page, dedication, and other one-page preliminaries (if you use them) are usually designed individually, as the information on them only occurs once in that format anywhere in the document.
- The table of contents and other related lists like figures and tables all need to share one design.
- Longer prelims like Foreword, Introduction, and Preface should likewise follow the same format between them.
- Chapter and Appendix start pages usually share a layout.
- Other (normal) pages have a single layout, but within the page there will be individual variations to handle tables, lists, figures, sidebars, exercises, footnotes, etc.

If you are going to design a whole document, it's probably a good idea to read a couple of books on layout design first, to get a feel for the conventions which contribute to making the reader comfortable reading.

While unusual or radical layouts have an important role in attention-grabbing, or in making a socio-political statement (*WIRED* magazine is an obvious recent example), they are usually out of place in business reports, white papers, books, theses, and journals. In ephemera, on the other hand, as in advertising, they are probably critical.

### 8.1.1 Spacing

We mentioned in § 7.3 and elsewhere the existence of the `geometry` package which lets you change margins. It also lets you set the text-area height and width and a lot of other layout settings: read the documentation for details (see § 5.1.2 for how to read package documentation).

```
\usepackage[left=2cm,top=1cm,bottom=2cm,right=3cm,  
nohead,nofoot]{geometry}
```

Bear in mind when using the `geometry` package that you only need to specify some of *either* the margins *or* the text height/width. Because it knows the paper size already, if you give it the text width and the left-hand margin, for example, it can work out the right-hand margin. A new feature introduced in 2010 is the `\newgeometry` command, which lets you reset the margin settings in mid-document. This isn't something you want to do very often, but until now it had been very difficult to do.

The spacing around the individual textual components (headings, paragraphs, lists, footnotes, etc.) can also be changed on a document-wide basis,

as we saw with paragraph spacing and indentation in § 3.6. There are a lot of packages available to do various aspects of this, far too many to list here: search CTAN<sup>1</sup> to find what you need.

Changing the spacing of section headings for the whole document can be done with the `sectsty` or `section` packages, designed to let you adjust section-head spacing without having to know about the internal  $\LaTeX$  coding, which is quite complex.

The spacing for lists can be adjusted with the `mdwlist` package. In both cases the user with highly specific requirements such as a publisher's Compositor's Specification should read the relevant sections in the *The  $\LaTeX$  Companion* or ask for expert help, as there are many internal settings which can also be changed to fine-tune your design, but which need some knowledge of  $\LaTeX$ 's internals.

All the above are for automating changes so that they occur every time in a consistent manner. You can also make manual changes whenever you need:

**Flexible vertical space** There are three commands `\smallskip`, `\medskip`, and `\bigskip`. These output flexible (dynamic, or 'rubber') space, approximately 3pt, 6pt, and 12pt high respectively, and they will automatically compress or expand a little, depending on the demands of the rest of the page (for example to allow one extra line to fit, or a heading to be moved to the next page without anyone except a typographer noticing the change). *These commands can only be used after a paragraph break* (a blank line or the command `\par`).

**Fixed vertical space** For a fixed-height space which will *not* stretch or shrink, use the command `\vspace` followed by a length in curly braces, e.g. `\vspace{18pt}` (again, this has to be after a paragraph break). Bear in mind that extra space which ends up at a page-break when the document is formatted *will get discarded entirely* to make the bottom and top lines fall in the correct places. To force a vertical space to remain and be taken into account even after a page break (very rare), use the starred variant `\vspace*`, e.g. `\vspace*{19pt}`.

**Double spacing** Double-spacing normal lines of text is usually A Bad Idea, as it looks very ugly. It is still unfortunately a requirement in some universities for thesis submission, a historical relic from the days of typewriters. Nowadays,  $1\frac{1}{3}$  or  $1\frac{1}{2}$  line spacing is considered acceptable, according to your font size. If your institution still thinks they should have double line spacing, they are probably wrong, and just don't understand that the world has moved on since the

<sup>1</sup> <http://ctan.org/search.html>

typewriter. Show them this paragraph and explain that they need to enter the 21st century and adapt to the features of computer typesetting. If they still insist, use the `setspace` package, which has commands for double line-spacing and one-and-a-half line spacing, but be aware that it will not look pretty.

The space between lines is defined by the value of the length variable `\baselineskip` multiplied by the value of the `\baselinestretch` command. In general, *don't meddle with `\baselineskip` at all*, and with `\baselinestretch` only if you know what you are doing. (Both can, however, safely be used as reference values in commands like `\vspace{\baselineskip}` to leave a whole line space.)

The value of `\baselineskip` changes with the font size (see § 8.2.4) but is conventionally set to 1.2 times the current nominal font size. This is a value derived from long experience: only change it if you understand what it means and what effect it will have.

Quite separately, there are some perfectly genuine and normal reasons for wanting wide line spacing, for example when typesetting a proof of a critical or variorum edition, where editors and contributors are going to want to add notes manually by writing between the lines, or where the text is going to be overprinted by something else like Braille, or in advertising or display text for special effects.

**Horizontal space** There is a horizontal equivalent to the `\vspace` command: `\hspace`, which works in the same way, so `\hspace{1in}` will insert a 1" space like this  in mid-paragraph. There are also some predefined (shorter) spaces available:

- `\thinspace` ( $\frac{1}{6}$  em), which we saw between single and double quotes in the last paragraph of § 2.5. It's also sometimes used between the full point after abbreviations and a following number, as in page references like p. 199, where a word space would look too big, and setting it solid would look too tight.
- `\enspace` ( $\frac{1}{2}$  em). There is no direct equivalent predefined in  $\text{\LaTeX}$  for 'mid' and 'thick' spaces as used by metal typesetters, although it would be possible to define them. The en as a unit is used as the width of a single digit in many fonts, as a convenience so that tables of figures are easy to line up.
- `\quad` (1 em).
- `\qqquad` (2 em).

Beyond this, all horizontal space within paragraphs is automatically flexible, as this is what  $\text{\LaTeX}$  uses to achieve justification. Never be

tempted to try and change the spacing between letters unless you have some professional training in typography. Some systems use inter-letter spacing (incorrectly called ‘tracking’) as an aid to justification and *it is almost always wrong to do so* (and looks it). While it is of course possible to change letterspacing in  $\text{\LaTeX}$  (with the `soul` package), it should only be done by a typographer, and then only very rarely, as the settings are very subtle and beyond the scope of this book.<sup>2</sup>

### 8.1.2 Headers and footers

$\text{\LaTeX}$  has built-in settings to control the page style of its default page layouts. These are implemented with the `\pagestyle` command, which can take one of the following arguments.

`plain` for a page number centered at the bottom — this is the default;

`empty` for nothing at all, not even a page number — use this when you are doing one-page documents like posters or handouts, where a page number is not meaningful;

`headings` for running heads based on the current chapter and section — this is common for articles, books, and reports, so that every page is identifiable even if printed or copied separately;

`myheadings` which lets you use your own [re]programmed definitions of how to use the `\markright` and `\markboth` commands, which control how chapter and section titles get into page headers.

The command `\thispagestyle` (taking the same arguments) can be used to force a specific style for the current page only.

However, the easiest way to get specialist running heads is to use the `fancyhdr` package, which lets you redefine the left-hand, centre, and right-hand page headers and footers for both odd-numbered and even-numbered pages (twelve objects in all). These areas can contain a page number, fixed text, variable text (like the current chapter or section title, or the catch-words of a dictionary), or even a small image. They can also be used to do page backgrounds and frames, by making one of them the top corner of an invisible box which ‘hangs’ text or images down over the whole page.

The settings for the typeset version of this document can be used as an example: for the whole story you have to read the documentation.

<sup>2</sup> This does not apply for the German technique in blackletter type of using letter-spacing instead of (non-existent) italics. The defaults in the `soul` package were designed to cater for this.

```

\pagestyle{fancy}\fancyhead{}
\renewcommand\headrulewidth{.1pt}
\fancyhead[LO,RE]{\footnotesize\sffamily\lite\leftmark}
\fancyhead[LE,RO]{\footnotesize\sffamily\lite\itshape
\rightmark}
\fancyfoot[C]{}
\fancyfoot[LE,RO]{\setlength{\fboxsep}{2pt}\ovalbox%
{\footnotesize\sffamily\thepage}}
\fancyfoot[LO,RE]{\footnotesize\sffamily\lite@title}
\fancypagestyle{plain}{\fancyhf{}
\fancyfoot[R]{\setlength{\fboxsep}{2pt}\ovalbox{%
\footnotesize\sffamily\thepage}}
\fancyfoot[L]{\footnotesize\sffamily\lite@title}
\renewcommand{\headrulewidth}{0pt}}

```

This is probably more complex than most documents, but it illustrates some common requirements:

1. Settings are prefixed by making the `\pagestyle` ‘fancy’ and setting the `\fancyhead` to null to zap any predefined values.
2. The thickness of the rule at the top of the page can be changed (or set to 0pt to make it disappear).
3. The header and footer settings are specified with L, C, and R for left, centre, and right; and with O and E for Odd and Even numbered pages. In each setting, the typeface style, size, and font can be specified along with macros which implement various dynamic texts (here, the current chapter and section titles, which  $\LaTeX$  stores in `\rightmark` and `\leftmark`).
4. The ‘plain’ variant is used for chapter starts, and resets some of the parameters accordingly.

## 8.2 Using fonts

The default typeface in  $\LaTeX$  is Computer Modern (CM). This typeface was designed by Knuth for use with  $\TeX$  because it is a book face, and he designed  $\TeX$  originally for typesetting books. Because it is one of the very few book typefaces with a comprehensive set of fonts, including a full suite of mathematics, it has remained the default, rather than the Times you find in wordprocessors, because until recently the mathematical symbols for Times were a commercial product often unavailable to users of free software.

Computer Modern is based on a 19th-century book typeface from Monotype, which is why it looks a little like an old-fashioned school book. This paragraph is set in Computer Modern so you can see what it looks like. The typeface was designed using METAFONT, the font-drawing program made by Knuth to accompany T<sub>E</sub>X systems, but it is now also available in Type 1 and TrueType formats.

### Web browser fonts



If you are reading this in a web browser, the above paragraph is only a low-resolution copy because browsers don't usually have the Computer Modern font available.

In addition to CM, there are many other METAFONT fonts which can be downloaded from CTAN, including a large collection of historical, symbol, initial, and non-Latin fonts. L<sup>A</sup>T<sub>E</sub>X also comes with the 'Adobe 35' typefaces which are built into laser printers and other DTP systems, and some more fonts donated by the X Consortium. Plus, of course, standard L<sup>A</sup>T<sub>E</sub>X can use any of the thousands of Type 1 fonts available, and *pdfL<sup>A</sup>T<sub>E</sub>X* can use any of the thousands of TrueType fonts as well.

X<sub>Y</sub>L<sub>A</sub>T<sub>E</sub>X



Jonathan Kew's X<sub>Y</sub>L<sub>A</sub>T<sub>E</sub>X (see the Preface on p. xiii) attempts to provide transparent access to any font of any format on your computer, making conversion and separate font installation unnecessary.


In the following lists, if there is a package available, its name is given in parentheses after the name of the typeface. The font-family name is shown on the right-hand side. If a non-standard font-encoding is needed, its name is shown before the font-family name.

### Latin-alphabet typefaces (METAFONT)

Computer Modern Roman	The quick brown fox jumps over the lazy dog	cmr
Computer Modern Sans	The quick brown fox jumps over the lazy dog	cmss
Computer Modern Typewriter	The quick brown fox jumps over the lazy dog	cmtt
Pandora (pandora)	The quick brown fox jumps over the lazy dog	panr
Pandora Sans	The quick brown fox jumps over the lazy dog	pss
Pandora Typewriter	The quick brown fox jumps over the lazy dog	pntt

The quick brown fox jumps over the lazy dog		
Universal		uni
The quick brown fox jumps over the lazy dog		
Concrete (ccr)		ccr
The quick brown fox jumps over the lazy dog		
Éireannach (eiad)		eiad
Ἡὺ λον τιντεῶν μαρι το εἰντεῶν πέιν		
Rustic		rust
THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG		
Uncial		uncl
The quick brown fox jumps over the lazy dog		
Dürer		cdr
THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG		
Fraktur		U yfrak
Fuchs, Du hast die Gans gestohlen, gib sie wieder her!		
Gothic		U ygoth
If it plese any man (pírituel or temporel		
Schwäbische		U yswab
Fuchs, Du hast die Gans gestohlen, gib sie wieder her!		

### Latin-alphabet typefaces (PostScript Type 1) from Adobe

Avant Garde (avant)		pag
The quick brown fox jumps over the lazy dog		
Bookman <sup>3</sup> (bookman)		pbk
The quick brown fox jumps over the lazy dog		
Courier (courier)		pcr
The quick brown fox jumps over the lazy dog		
Helvetica (helvet)		phv
The quick brown fox jumps over the lazy dog		
New Century Schoolbook <sup>4</sup> (newcent)		pnc
The quick brown fox jumps over the lazy dog		
Palatino <sup>5</sup> (palatino)		ppl
The quick brown fox jumps over the lazy dog		
Symbol		U psy
Τηε θυιχκ βρωων φοξ φυμπσ οωερ τηε λαζψ δογ		
Times New Roman <sup>6</sup> (times)		ptm
The quick brown fox jumps over the lazy dog		
Zapf Chancery (zapfchan)		pzc
<i>The quick brown fox jumps over the lazy dog</i>		
Zapf Dingbats (pifont)		U pzd
		

<sup>3</sup> Uses Avant Garde as the sans-serif and Courier for monospace.

<sup>4</sup> Uses Helvetica as the sans-serif font and Courier for monospace.

<sup>5</sup> Uses Avant Garde as the sans-serif and Courier for monospace.

<sup>6</sup> Uses Helvetica as the sans-serif font and Courier for monospace. Mathematical symbols for Times are available both free and commercially.



As mentioned in §4.4, the ‘Adobe 35’ fonts can be used with any printer, not just a laser printer or typesetter. The *Ghostscript* interpreter and the *GSview* viewer come with a large set of printer drivers, so you just create PostScript output and print from *GSview*.

Incidentally, the 35 refers to the total number of fonts for the 10 typefaces, including their bold, italic, and bold-italic variants.

Postscript Type 1 fonts have been the mainstay of the graphic arts industries for many years, as they allow much better definition of variance (‘hinting’) than most other formats. However, the font format remains proprietary to Adobe, even though they have released it for public use, which means they could change it without warning. A new format called ‘OpenType’ is designed to overcome this, and some versions of  $\TeX$  are already able to use OpenType fonts.

### Latin-alphabet fonts (PostScript Type 1) from the X Consortium

Charter (charter)	The quick brown fox jumps over the lazy dog	bch
Nimbus Roman	The quick brown fox jumps over the lazy dog	unm
Nimbus Sans	The quick brown fox jumps over the lazy dog	unms
URW Antiqua	The quick brown fox jumps over the lazy dog	uaq
URW Grotesk	<b>The quick brown fox jumps over the lazy dog</b>	ugq
Utopia <sup>7</sup> (utopia)	The quick brown fox jumps over the lazy dog	put

### Non-Latin-alphabet typefaces (METAFONT)

Cypriot	↓Λ* @Υ*Υ↑ ωΩ≅Χ† +≅)( 0Υ*†V ≅*ΜΩ †Λ* V*V ≅≅>X	cypr
Etruscan	The qik bron fox mps over the lazy dog	etr
Linear ‘B’	⊕ϕΑ ϕϕΥ∇⊕ †L⊕M̄ †⊕M ⊕ϕ⊕†Υ ⊕⊕AL ϕϕA †ϕ∇ †⊕†	T1 linb
Phoenician	⊗⊕⊕ ϕϕ ϕ†OY† FO† ††⊕ ⊕⊕† †⊕⊕ †>I† ΔOT	phnc
Runic	†HM NIL BR††† †RY †NMK† RM††† †HM ††† MXX	fut
Bard	†N 4V << ††⊕†† †⊕†† †VW†† ⊕†††† †N †††† >⊕†	U zba

You should note that these are just the defaults installed with all versions of  $\TeX$ . There are hundreds more listed in Palle Jørgensen’s comprehensive

<sup>7</sup> Removed from recent distributions as it is not free.

## What types of font?



Just to make it clear: standard  $\LaTeX$  uses only METAFONT and PostScript Type 1 fonts. pdf $\LaTeX$  can use TrueType fonts as well, but needs special configuring to do so. X $\LaTeX$  can use almost any type of font you have installed.

$\LaTeX$  Font Catalog published by the Danish  $\TeX$  Users Group at <http://www.tug.dk/FontCatalogue/>.

### 8.2.1 Changing the default font family

$\LaTeX$  expects to work with three font families as defaults:

Font family	Code
Roman (serif, with tails on the uprights), the default	<code>rm</code>
Sans-serif, with no tails on the uprights	<code>sf</code>
Monospace (fixed-width or typewriter)	<code>tt</code>

The start-up default for  $\LaTeX$  equates the `rm` default with the `cmr` font-family (Computer Modern Roman), `sf` with `cmss`, and `tt` with `cmtt`. If you use one of the packages listed in the tables on pp. 133–135, it will replace the defaults of the same type: for example, the `bookman` package makes the default `rm` font-family Bookman (`pbk`), but leaves the sans-serif (`sf`) and monospace (`tt`) families untouched. Equally, the package `helvet` changes the default sans-serif family to Helvetica<sup>8</sup> but leaves the serif (Roman) and monospace families untouched. Using both commands will change both defaults because they operate independently.

*However...* as it is common to want to change all three defaults at the same time, some of the most common ‘suites’ of typefaces are provided as packages, but they are for text work only, as they leave mathematics in Computer Modern:

`times` changes to Times/Helvetica/Courier.

<sup>8</sup> The Helvetica typeface family has a notoriously large x-height, making it hard to match with other typefaces at the same nominal size. The `helvet` package therefore has a `scaled` option that lets you reduce the optical size slightly so that the font sits more easily with others: `\usepackage[scaled=0.86]helvet`, for example.

`pslatex` same as `times` but uses a specially narrowed Courier to save space (normal Courier is rather inelegantly wide). This is the preferred setting if you want Times.<sup>9</sup>

`newcent` changes to New Century Schoolbook/Helvetica/Courier.

`palatino` changes to Palatino/Avant Garde/Courier.

If you use mathematics, there are two fairly complete implementations of non-CM typefaces in the `mathptmx` (Times) and `mathpazo` (Palatino) packages. The whole business of math fonts is perpetually under revision in any case, as mathematicians are perpetually inventing new symbols, which take a while to appear in typefaces. The American Mathematical Society (AMS) and other organisations are involved with a project called Styx, which is expected eventually to define the complete suite of mathematical characters in a rational and extensible manner — but don't hold your breath.

Where no package name is given in the tables on pp. 133–135, it means the font is rarely used as a default by itself except in special cases like users' own homebrew packages. To use such a font you have to specify it manually, or make a little macro for yourself if you use it more than once.

### 8.2.2 Changing the font-family temporarily

To shift to another font family on a temporary basis, use the commands `\fontencoding` (if needed), `\fontfamily`, and `\selectfont`, and *enclose the commands **and** the text in curly braces*.

#### Grouping

This is a different way of using curly braces to how we have used them before: see the panel 'Grouping' on p. 138 — It limits the effect of a simple command (see § 2.3.1) to the material inside the braces.

In this example, the `\fontencoding` command has been used to ensure that the typeface will work even if the sentence is used in the middle of something typeset in a different encoding (like this document).<sup>10</sup>

In a normal document, of course, random typeface changes like this are rather rare. You select your typeface[s] once at the start of the document, and stick with them.

<sup>9</sup> The `pslatex` package is also said to be outdated by some experts because it implements rather long-windedly what can now be done in three commands. However, until these replace the current version, I recommend continuing to use `pslatex` when you want Times with Helvetica and narrow Courier.

<sup>10</sup> Test for the observant reader: in what typeface will the colon (:) in the example be set?

```
{\fontfamily{phv}\selectfont
Helvetica looks like this}:
{\fontencoding{OT1}\fontfamily{bch}\selectfont
Charter looks like this}.
```

Helvetica looks like this: Charter looks like this.

## Grouping



Note carefully this use of curly braces to restrict the scope of a change rather than delimit the argument to a command. This is called a **group**, and it makes the effect any changes made inside the braces local, so that they do not interfere with the text following. Any changes to fonts or other values made within the curly braces cease when the closing curly brace is processed. If you use a paragraphic formatting command like `\centering`, `\flushleft`, or `\flushright` in a group, you must end the text with a `\par` to finish the paragraph, otherwise the formatting will not take effect. If you use an environment like `center`, `flushleft`, or `flushright`, you do not need the `\par` because environments are inherently paragraphic and will do it for you.

Most cases where people want to do unusual typeface changes involve things like special symbols on a repetitive basis, and  $\text{\LaTeX}$  provides much easier programmable ways to make these changes into shorthand commands (called macros: see Chapter 9). You could, for example, make a macro called `\product` which would let you typeset product names in a distinct typeface:

```
Andlinger, Inc., has replaced \product{Splosh}
with \product{SuperSplosh}.
```

This is one of  $\text{\LaTeX}$ 's most powerful features. It means that if you needed to change your `\product` command at some later stage to use a different font, you only have to change three characters in the macro (the font-family name), and you don't need to edit your document text at all! What's more, a macro could do other things at the same time, like add an entry to an index of products.

Table 8.1: Typeface styles, families, shapes, and series

Type style	Command	Example (using Computer Modern)
Upright	<code>\upshape*</code>	The quick brown fox jumps over the lazy dog
Italic	<code>\itshape</code>	<i>The quick brown fox jumps over the lazy dog</i>
Slanted	<code>\slshape*</code>	<i>The quick brown fox jumps over the lazy dog</i>
Small Capitals	<code>\scshape*</code>	THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG
Bold	<code>\bfseries*</code>	<b>The quick brown fox jumps over the lazy dog</b>
Bold Extended	<code>\bfseries†</code>	<b>The quick brown fox jumps over the lazy</b>
Sans-serif	<code>\sffamily</code>	The quick brown fox jumps over the lazy dog
Monospace	<code>\ttfamily</code>	The quick brown fox jumps over the lazy dog

\* Not all typefaces have all variants! Some only have bold and italics.

† Some typefaces do not have both bold and bold extended: by default  $\LaTeX$  uses `\bfseries` for bold extended.

However, vastly more common are changes to type *style*, while staying with the same font-family.

### 8.2.3 Changing font style

Within each typeface or font family there are usually several different ‘looks’ to the type design.  $\LaTeX$  distinguishes between *font family*, *font shape*, and *font series* (see Table 8.1).

#### Commands without arguments

Remember back in §2.3.1 seeing simple commands with no arguments? The commands above affect all following text, so to restrict their effect, group them in curly braces along with the text you want affected.

These ‘shape’, ‘series’, and ‘family’ commands are *commutative*, so you can combine a shape with a series and/or a family, without the need to use `\selectfont`:

```
This gives you {\bfseries\itshape\sffamily bold italic
sans-serif type}, but beware
```

This gives you **bold italic sans-serif type**, but beware of pushing your fonts beyond their limits unless you are a typographer. It is not normally meaningful to combine one shape or series class with another of the same

class, such as trying to get slanted-italics. It's an impossibility to combine one family with another (such as a seriffed sans-serif typeface!). Slanted plus italics, for example, doesn't make sense, as italics are already slanted (although it is technically possible); and while some typefaces may well possess italic small caps, they are not in common use. Sans-serif and monospace (typewriter) are different fonts, and often different typeface families entirely.<sup>11</sup>

There is an alternative syntax for the most common type shape and series commands which uses curly braces in the normal 'argument' manner:

Type style	Command	Example
Italic	<code>\textit{text}</code>	puts <i>text</i> into italics
Slanted	<code>\textsl{text}</code>	puts <i>text</i> into slanted type*
Small Capitals	<code>\textsc{text}</code>	puts <b>TEXT</b> into small caps
Bold	<code>\textbf{text}</code>	puts <b>text</b> into bold type
Sans-serif	<code>\textsf{text}</code>	puts text into sans-serif type
Monospace	<code>\texttt{text}</code>	puts text into typewriter type

\* If slanted is available separately from italics.

You can nest these inside one another too:

```
...\textbf{\itshape\textsf{bold italic sans-serif type}}...
```

Underlining isn't a font, and it is extremely rare in typography except for special purposes. If you think you need it, use the `ulem` package with the `normalem` option, and the `\uline` command.

### 8.2.4 Font sizes

LaTeX has built into its defaults a set of predefined font size steps corresponding more or less to the traditional sizes available to metal typesetters. This is deliberate, as these sizes have grown up over 500 years of printing as those which go best together for book-work, which is where TeX originated.

These sizes are also reflected in the size steps at which Computer Modern was designed. It often comes as a surprise to new users that many typefaces are not designed as a single font and just scaled up or down, but specially drawn at different sizes to make them more legible.

<sup>11</sup> Although if you're a typographer wanting to experiment with typewriter typefaces with and without serifs, you can use METAFONT to do exactly this kind of thing. But that's way outside the scope of this document.

As an example, here's 12pt Computer Modern, and here's 5pt Computer Modern scaled up to 12pt, and here's 17pt Computer Modern scaled down to 12pt so you can see there really is a significant difference. In general, you probably don't want to go scaling fonts too much beyond their design size because they will start to look very odd.

The default sizes (and the commands that operate them) are based on the use of a 10pt font, which is the normal size for most texts. Using the larger defaults (11pt and 12pt) for the body font will use 11pt and 12pt designs, with other sizes (eg headings) resized to match. The exact sizes used are listed in the macros in the Class Option files `size10.clo`, `size11.clo` and `size12.clo`. TeX's default fonts above 10pt are in fact scaled by a factor of 1.2, as shown in the fourth column of the table below.

Command	Example	Nominal point size	Exact point size
<code>\tiny</code>	The quick brown fox jumps over the lazy dog	5	5
<code>\scriptsize</code>	The quick brown fox jumps over the laz	7	7
<code>\footnotesize</code>	The quick brown fox jumps over the l	8	8
<code>\small</code>	The quick brown fox jumps over th	9	9
<code>\normalsize</code>	The quick brown fox jumps over	10	10
<code>\large</code>	The quick brown fox jumps	12	12
<code>\Large</code>	The quick brown fox ju	14	14.40
<code>\LARGE</code>	The quick brown fo	18	17.28
<code>\huge</code>	The quick brown	20	20.74
<code>\Huge</code>	The quick bro	24	24.88

#### Commands without arguments (again)

The commands above affect all following text, so to restrict their effect, group them in curly braces along with the text you want affected. The inter-line spacing gets reset in proportion, but only gets applied at the end of the paragraph, so if you have a whole paragraph or list item to be set in Large type, there needs to be a `\par` before the closing curly brace, otherwise the default line-spacing will be used.

While these 'shorthand' commands relieve the beginner of having to worry about the 'right' size for a given task, when you need a specific size you can use the `fix-cm` package to override the step sizes, and use the `\fontsize` command:

```
\fontsize{22}{28}\selectfont This is 22pt type 6pt leaded
```

‘Leading’ comes from the old metal-type practice of adding a lead strip between lines to increase the spacing.

### Ordinal superscripts



Don’t use them in normal text. Superscripted ordinals (like 21<sup>st</sup>) are a historical relic of Victorian typography. They are unnecessary and are never used in modern professional typesetting. They were re-introduced by Microsoft Word apparently because some Americans like their wordprocessing to look old-fashioned.

If you want to try and mimic low-quality wordprocessing, or if you are trying to make a typographic fac-simile of Victorian typesetting, use the `\textsuperscript` command from the `textcomp` package. Do not use math mode for text superscripts.

The `\fontsize` command takes two arguments: the point size and the baseline distance. The above example gives you 22pt type on a 28pt baseline (i.e. with 6pt extra space or ‘leading’ between the lines).

### Step-sized fonts



Computer Modern fonts (the default) come fixed at the named size steps shown in the table, and if you try to use an odd size in between,  $\LaTeX$  will pick the closest step instead. Many other fonts follow this pattern: if you need to use fonts at arbitrary other sizes there is a package `fix-cm` which lets you override the default steps.

## 8.2.5 Logical markup

All this playing around with fonts is very pretty but you normally only do it for a reason, even if that reason is just to be decorative. Italics, for example, are used for many things:



Cause	Effect
Foreign words	<i>ex officio</i>
Scientific names	<i>Ranunculus ficaria</i>
Emphasis	<i>must not</i>
Titles of documents	<i>The <math>\text{\LaTeX}</math> Companion</i>
Product names	Corel's <i>WordPerfect</i>
Variables in maths	$E = mc^2$
Subtitles or headings	<i>How to get started</i>
Use of a letter as a word	Who knocked the <i>L</i> out of London?
Decoration	<i>FREE UPGRADE!!!</i>

Humans usually have no problem telling the difference between these reasons, because they can read and understand the meaning and context. Computers cannot (yet), so it has become conventional to use descriptive names which make the distinction explicit, even though the appearance may be the same.

$\text{\LaTeX}$  has some of these built in, like `\emph`, which provides *emphasis*. This has a special feature because *when the surrounding text is already italic, emphasis automatically reverts to upright type*, which is the normal practice for typesetting.

This has a special feature because `{\itshape when the surrounding text is already italic, \emph{emphasis}}` automatically reverts to upright type, which is the

This sensitivity to logic is programmed into the definition of `\emph` and it's not hard to make up other commands of your own which could do the same, such as `\foreign` or `\product`.

But why would you bother? In a short document it's probably not important, but if you're writing a long report, or a formal document like an article, a book, or a thesis, it makes writing and editing hugely easier if you can control whole groups of special effects with a single command, such as italicising, indexing, or cross-referencing to a glossary. If a format needs changing, you only have to change the definition, and every occurrence automatically follows suit.

It also makes it possible to find and act on groups of meanings — such as making an index of scientific names or product names (as in this document) — if they are identified with a special command. Otherwise you'd spend weeks hunting manually through every `\textit` command to find the ones you wanted. This is the importance of automation: it can save you time and money.

**Warning from the past**

Beware of this 'vaine conceipt of simple men, which judge things by ther effects, and not by ther causes'. (Edmund Spenser, 1633)

It's hugely more efficient to have control of the cause than the effect.

In Chapter 9 we will see how to make your own simple commands like this.

**8.2.6 Colour**

You can typeset anything in  $\text{\LaTeX}$  in any colour you want using the `xcolor` package. First, you need to add the command `\usepackage{xcolor}` to your preamble (note the US spelling of color!). This makes available a default palette of primary colours: `red`, `green`, and `blue` for the RGB colour model used for emitted light (computer and television screens), and `cyan`, `magenta`, `yellow`, and black for the CMYK colour model used for reflected light (printing).

For the occasional word or phrase in colour, use the command `\textcolor` with two arguments, the colour name and the text:  
`\textcolor{red}{like this}`. There is a `\color` command as well, for use within groups:

```
...{\color{blue}some text in blue}...
```

If you have the PostScript printer driver `dvips` installed, `xcolor` provides a 64-colour palette of predefined *color names*. These represent approximately the colours in the big box of 64 *Crayola* colouring pencils much favoured by artists and designers. To get this palette, use `\usepackage[dvipsnames]xcolor`.

Now if you want the *Crayola* colour `Crimson`, you can use it as a colour name:

```
\color{Crimson}
\textcolor{Crimson}{some red text}
```

The biggest selection of predefined colours is available with the `svgnames` option: this defines all the colours defined by the Scalable Vector Graphics (SVG) standard for web graphics (so it's good for compatibility).

As some of the predefined colour names are quite long, you can create a short name of your own for colours you use frequently, using the `\definecolor` command:

```
\definecolor{mb}{named}{MidnightBlue}
```

The `\definecolor` command needs three arguments: your shorthand name, the name of the colour model, and the colour specification. In the case of the `named` model, the last argument is one of the colour names specified by the option you loaded the package with. To use these names with *pdf $\LaTeX$* , you may need to use the `pdftex` option to the `xcolor` package, depending on the version of  $\TeX$  you have installed.

Using the `\definecolor` command, you can define any colour you want by giving it a name, specifying which colour model, and providing the Red-Green-Blue (RGB) or Cyan-Magenta-Yellow-Black (CMYK) colour values *expressed as decimal fractions, separated by commas*. For example, an RGB shade given as (37,125,224) in decimal (`#250FE0` in hexadecimal as used on the Web) can be given as:

```
\definecolor{midblue}{rgb}{0.145,0.490,0.882}
```

(To get the fractional value, divide the integer decimal value by 255, the maximum for each of the hues in the Red-Green-Blue colour model.) You can then use `\textcolor` with your new colour name: `midblue` looks like `this` if you're reading in colour.

The `xcolor` package also provides a colour version of `\fbox` (see § 6.7.2) called `\colorbox`:

```
\colorbox{midblue}{\color{magenta}Magenta on midblue}
```

However, combining colours is an art and a skill: using conflicting colours like `Magenta on midblue` is as good a warning as any to learn about colour models and palettes before trying to use them!

## 8.3 Installing new fonts

Different fonts come in a variety of packagings: the three most common used with  $\TeX$  systems are PostScript fonts, TrueType fonts, and METAFONT fonts. How you install them and where they go depends on

## Directories (Folders)



In the examples below, all the folders (directories) are assumed to be in your Local  $\TeX$  Directory unless otherwise explicitly given. See Chapter 1 for how to find this out.

how you installed  $\LaTeX$ : all I can deal with here are the standard locations within the TDS.

Typefaces come supplied as one or more font ‘outline’ files and a number of ancillary files:

**METAFONT typefaces** have a number of `.mf` source (outline) files, possibly also some `.fd` (font definition) files and a `.sty` (style) file. The `.tfm` ( $\TeX$  font metric) files are not needed, as they can be generated from the outlines.

**PostScript typefaces** come as a pair of files: a `.pfb` (PostScript font binary) or `.pfa` (PostScript font ASCII) outline, and an `.afm` (Adobe font metric) file. There may also be `.inf` and other files but these are not needed for use with  $\TeX$  systems.

**TrueType typefaces** are a single `.ttf` file, which combines outline and metrics in one.

The instructions here assume the use of the New Font Selection Scheme (NFSS) used in  $\LaTeX 2_{\epsilon}$ . If you are running the obsolete  $\LaTeX 2.09$ , upgrade it now.

### 8.3.1 Installing METAFONT fonts

This is the simplest installation. When you download METAFONT fonts from CTAN, you’ll usually find a number of outline files (`.mf` files) and maybe some other types as well (see below).

1. Create a new subdirectory named after the typeface you’re installing in `fonts/source/public/`:
2. Copy all the `.mf` files to this directory.
3. Copy the `.fd` (Font Definition) file[s] to your `tex/latex/mfnfss` directory.
4. Copy the `.sty` (style) file to a subdirectory (create it) of `tex/latex` named after the typeface.

5. Run your  $\TeX$  indexer program (see step 4 in the procedure on p. 82).

That's it. Unlike PostScript fonts, METAFONT fonts can be used to generate the font metric file (.tfm files) automatically on-the-fly the first time the typeface is used, so there should be nothing else to install.

Now you can put a `\usepackage` command in your preamble with whatever name the .sty file was called, and read the documentation to see what commands it gives to use the font (refer to the last paragraph of § 5.2.1 and step 2 in the procedure on p. 81).

If the font came *without* .fd or .sty files, you'll need to find someone who can make them for you (or follow the outline in § 8.3.2, step 11 in the procedure on p. 153).

### 8.3.2 Installing PostScript fonts

Lots of people will tell you that PostScript fonts and PostScript output are dead and that TrueType or OpenType fonts and PDF output are the way to go. While this may be true for some cases, standard  $\TeX$  does not work with TrueType fonts and does not produce PDF directly. Only *pdf $\TeX$*  does that, and there are still many printers whose typesetters and platemakers use PostScript equipment rather than PDF. In addition, operating system support for scalable fonts is still very poor on Unix systems (including GNU/Linux), despite the advances in recent years, and many rebranded ('knock-off' or pirated) TrueType fonts supplied with other systems are of very poor quality, so in many cases it still makes sense to use  $\TeX$ 's built-in support for PostScript. When  $\X\TeX$  becomes the default processor, it will no longer be necessary to install different types of font files separately, as  $\X\TeX$  is able to use all fonts natively from your system's font folder.

Two files are needed for each font: the .afm Adobe Font Metric (AFM) and the .pfb PostScript Font Binary (PFB) files. *You must have both for each font before you start.* If you only have the near-obsolete .pfa PostScript Font ASCII (PFA) files, it may be possible to generate the .pfb files using the *t1binary* program from the *t1utils* suite (see <http://gnuwin32.sourceforge.net/packages/t1utils.htm>) or the excellent *FontForge* font editor (from <http://fontforge.sourceforge.net>). There are unfortunately still some companies distributing Type 1 fonts in .pfa format (Mathematica is one reported recently).

I'll repeat this: before you start, make sure you have all the .afm and .pfb files for the typeface you want. In the example below, I'm going to use a single font from an imaginary typeface called Foo, so I have foo.afm and foo.pfb files.

#### 1. Put the files in your temporary directory

This is /tmp on Macs and GNU/Linux, and should be C:\tmp or C:\temp or even C:\Windows\temp on Microsoft Windows.

## 2. Decide on the short font name to use inside $\text{\LaTeX}$ .

This is *not* the full descriptive name (e.g. Baskerville Italic Bold Extended) but an encoded font name in the format `fnnsssec`, devised by Karl Berry, which stores the same information in no more than eight characters for compatibility with systems which cannot handle long filenames. The letters in the format above have the following meanings (see the *fontname* documentation on your computer for more details — lists of the codes used are in the files `supplier.map`, `weight.map`, `width.map`, `variant.map`, and the various `.map` files for each foundry):

Letter	Meaning	Examples
f	foundry	b=Bitstream, m=Monotype, p=Adobe
nn	typeface	ba=Baskerville, tm=Times, pl=Palatino
ss	series/shape	r=roman, b=bold, i=italic, etc.
ee	encoding	8r=revised $\text{\TeX}$ Base1, 1y=Y&Y's $\text{\TeX}$ 'n'ANSI
v	variant	smallcaps, outline, script, etc.

The `fonts/fontname` directory in your main (*not* local) installation directory of  $\text{\TeX}$  has files for several foundries giving fully-formed names like these for common fonts (e.g. `ptmr8r` is [Adobe] PostScript Times Roman in an 8-bit revised  $\text{\TeX}$  encoding; `bgs1ly` is Bitstream Gill Sans Light in Y&Y's  $\text{\TeX}$ 'n'ANSI encoding [LY1]).<sup>12</sup> Read the documentation in *Fontname: Filenames for  $\text{\TeX}$  fonts* to find out how to make up your own short names if the foundry and font you want is not shown in the lists in the `fonts/map/fontname` directory.

In this example we'll call our mythical example typeface 'zork' (standing for Zfonts Ordinary Bookface, because `k` is the letter used for Book fonts, `b` being already the code for bold) and we'll assume the font comes in the two files `foo.afm` and `foo.pfb` that I mentioned above.

While the `font/map/fontname` directories have ready-made maps of these names for popular collections of typefaces, making them up requires some knowledge of typographic terms and a careful reading of the *fontname* documentation.

## 3. Decide on your encoding

Encoding is needed because Adobe fonts store their characters in

<sup>12</sup> Confusingly, Bitstream fonts (and others from similar sources) mostly have different names from the original fonts, to avoid copyright issues, so what they call Humanist 521 is actually Gill Sans. Until recently, US law only allowed the *names* of typefaces to be copyrighted, not the font designs themselves, leading to widespread piracy.

different places to the T<sub>E</sub>X standard. This is what tripped me up the first few times until someone pointed me at Y&Y's<sup>13</sup> T<sub>E</sub>X'n'ANSI encoding which at the time was the only one that included the glyphs I want where I expected them to be.<sup>14</sup> Now, however, I recommend using the 8r encoding for all PostScript fonts. The encoding vector file `8r.enc` should be in your main (*not* local) T<sub>E</sub>X installation directory in `fonts/enc/dvips/base`.

To avoid having to type the long path each time below, just copy this file to the temporary directory where you're doing all this stuff.

#### 4. Convert .afm files to .tfm

The `afm2tfm` and `vptovf` programs are standard T<sub>E</sub>X utilities in the `bin` directory of your main T<sub>E</sub>X installation.

In a command window, type:

```
afm2tfm foo.afm -v zorkly.vpl -p texnansi.enc rzorkly.tfm >zork.id
```

This creates a special 'raw' T<sub>E</sub>X Font Metric file (hence the special `r` prefix) that L<sup>A</sup>T<sub>E</sub>X can use, with a list of all its properties encoded with 8r (the `.vpl` or Virtual Property List file). Many people will tell you that virtual fonts are dead and that this is the wrong way to do it, but no-one has ever shown me an alternative that works, so I stick with it.

#### 5. Small caps (optional)

If you want a small caps variant faked up (perhaps because the typeface family doesn't have a special small-caps font), repeat the medicine like this:

```
afm2tfm foo.afm -V zorklyc.vpl -p texnansi.enc rzorkly.tfm >>zork.id
```

Note the `capitalV` option here. Yes, it *does* overwrite the `rzorkly.tfm` created in the first command. Let it. And those are *two* of the 'greater-than' signs before the `zork.id` filename because we want to append to it, not overwrite it.

<sup>13</sup> Y&Y, Inc has ceased trading and their T<sub>E</sub>X distribution is not longer available, although there is email support at <http://lists.ucc.ie/lists/archives/yandytex.html>, and their encoding files continue to be used.

<sup>14</sup> The only one I had problems with is  $\text{Å}$ , which for some weird reason isn't catered for in this encoding.

**6. Create the virtual font**

Turn the `.vpl` files into `.vf` and `.tfm` pairs.  $\text{\LaTeX}$  needs these to convert from Adobe's encoding to its own.

```
vptovf zorkly.vpl zorkly.vf zorkly.tfm
vptovf zorklyc.vpl zorklyc.vf zorklyc.tfm
```

**7. Make directories to hold the files**

Under your local directory there should be a `fonts` directory, and in it there should be `afm`, `tfm`, `type1`, and `vf` subdirectories. Create them if they do not already exist.

In each of these four, create a directory for the foundry, and within them create a directory for the typeface (using a human-readable typeface name, not the short Karl Berry fontname). On my computer, this means:

```
cd /usr/local/share/texmf/fonts mkdir -p afm/zfonts/zork
mkdir -p tfm/zfonts/zork mkdir -p type1/zfonts/zork
mkdir -p vf/zfonts/zork cd /tmp
```

Or if you're lazy:

```
(cd /usr/local/share/texmf/fonts;for d in afm tfm type1
vf; do mkdir -p $d/zfonts/zork;done)
```

For Microsoft Windows users, you may have to do this in the  $\text{\MikTeX}$  directories.

The `mkdir` command exists in Windows as well, but the `-p` is a Unix feature: it automatically creates any missing intervening subdirectories. If your directory-making command doesn't do this, you'll have to make the intervening directories by hand first.

**8. Copy the files to their rightful places**

Copy the four groups of files to the four new directories:

```
cp *.afm \textsl{\uline{localdir}}/fonts/afm/zfonts/zork/
cp *.tfm \textsl{\uline{localdir}}/fonts/tfm/zfonts/zork/
cp *.pfb \textsl{\uline{localdir}}/fonts/type1/zfonts/zork/
cp *.vf \textsl{\uline{localdir}}/fonts/vf/zfonts/zork/
```



where *localdir* is the prefix of your local T<sub>E</sub>X installation directory. You can of course do all this with a directory window and mouse if you find it easier.

### 9. Create a font map

The font map is what tells *dvips* which PFB file to use for which font.<sup>15</sup> There should be a personal configuration file for your use of L<sup>A</sup>T<sub>E</sub>X called `10local.cfg`, in your login folder in a subfolder called `.texmf-config/updmap.d`. All it needs is an entry for our new font, using the three-letter font family abbreviation (the first three letters of the Karl Berry fontname (here 'zor')):

```
Map zor.map
```

We also have to create this map file (`zor.map`) in a subdirectory of `dvips/config/` named after the foundry, so we need to create `dvips/config/zfonts` as well.

- (a) Open `10local.cfg` in your editor.
- (b) At the bottom, add the line: `Map zor.map`
- (c) Save and close the file.

The font entries in our `zor.map` will be on a *single* line each, with no line-wrapping. Each entry gives the short name of the font, the long (Adobe) name, the PostScript encoding parameters (in quotes), and then two filenames prefixed by input redirects (less-than signs): the encoding file and the PostScript outline file.

- (a) First create the directory if it doesn't already exist:

```
mkdir -p \textsl{\uline{localdir}}/dvips/config/zfonts
```

- (b) Use your editor to open (create) the file `dvips/config/zfonts/zor.map`.
- (c) Insert the line:

<sup>15</sup> The configuration file for *dvips* is `config.ps` and there are others for other drivers (e.g. PDF): they get their entries from the program *updmap* which reads map files for each typeface. The configuration file for *updmap* is `updmap.cfg`, and that in turn is updated by the program *updmap-sys*. But hopefully you won't need to know that.

```
rzorkly Zork-Blackface "TeXnANSIEncoding ReEncodeFont" <texnansi.enc <foo.pfb
```

- (d) Save and close the file.

You get the full font name (here, ‘Zork-Blackface’) from the `zork.id` which was created back in step 4 in the procedure on p. 149 when we ran `afm2tfm`. You must get this exactly right, because it’s the ‘official’ full name of the font, and PostScript files using this font need to match it.

#### 10. Create a style file

ℒ<sub>T</sub>E<sub>X</sub> needs a style file to implement the interface to the font. Call it after the typeface or something related; in this example we’ll call it `foozork.sty`. In it go some details of the name and date we did this, what version of ℒ<sub>T</sub>E<sub>X</sub> it needs, and any other command necessary to operate the font, like the font encoding and whether it is to supersede the current default Roman font.

- (a) Use your editor to open (create) `foozork.sty` in your local `tex/latex/psnfss` directory.
- (b) Insert the following lines:

```
% fozork - created from foo for Zork
\def\fileversion{1.0}
\def\filedate{2010/12/03}
\def\docdate{2010/12/03}
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{fozork}[\filedate\space\fileversion\space
Zfonts Ordinary PSNFSS2e package]
\RequirePackage[T1]{fontenc}
\renewcommand{\rmdefault}{zor}
\endinput
```

Note the following:

- The first argument to `\ProvidesPackage` *must* be the same as this style file name; and that the font family is referred to as `zor`, being the foundry letter plus the fontname abbreviation. This acts as a prefix for any/all font variants (bold, italic, etc.).
- The `T1` option to the `fontenc` package is the name used for the interface to the 8r encoding.

- If this is a typewriter font, make the renewed command `\rmdefault` into `\ttdefault`. If it's a sans-serif font, make it `\sfdefault` instead.
- Omit that command completely if you don't want the style file to supersede the current defaults but simply to make the font available. If you do this, you probably want to write a new command or two to use it, typically one for grouped use and one for argument use:

```
\newcommand{\zorkfamily}{\fontencoding{T1}\fontfamily{zor}\selectfont}
\newcommand{\textzork}[1]{\zorkfamily#1}
```

- (c) Save and close the file.

### 11. Create the Font Definition file

The last file to create is the *font definition* (.fd) file. This is named following the pattern `eefnn.fd`, using the same conventions as before, by prepending the (lowercase) encoding abbreviation to the foundry letter and fontname abbreviation, so our example would be `t1zor.fd` for the T1 (8r) encoding and the `zor` short font name.

- (a) Use your editor to open (create) `tex/latex/psnfss/t1zor.fd`  
 (b) Enter the following lines:

```
\ProvidesFile{t1zor.fd}[2010/03/03 v0.1 manual
    font definitions for LY1/zor.]

\DeclareFontFamily{T1}{zor}{}

\DeclareFontShape{T1}{zor}{k}{n}{<-> zorkly}{}
\DeclareFontShape{T1}{zor}{k}{sc}{<-> zorklyc}{}

```

- (c) Save and close the file.

FD files typically use one `\DeclareFontFamily` command which specifies the encoding and the short font name. Then as many pairs of `\DeclareFontShape` commands as you converted fonts (assuming you did both normal and small caps for each font: see step 5 in the procedure on p. 149; if you didn't, then only one such command per font is needed here). The arguments to the `\DeclareFontShape` command to watch are the 3rd (weight/width), 4th (shape), and 5th (font outline

name): the rest are static for each .fd file and simply identify the encoding and the font family.

The codes to use are given on pages 414–15 of the *The L<sup>A</sup>T<sub>E</sub>X Companion* and should also be in your copies of `fonts/map/fontname/weight.map` and `fonts/map/fontname/width.map`. The rules for combining weight and width need care: read the documentation for the `fontname` package. There is no `shape.map` in `fontname` because it's not part of font file names, it's purely a L<sup>A</sup>T<sub>E</sub>X creation, so here's what the same book says:

Character	Meaning
n	normal (upright)
it	italic
sl	slanted
sc	small caps
ui	upright italic
ol	outline

Add your own for other oddities, but be consistent: I use `cu` for cursive (scripts), for example, and `k` for blackletter faces (not to be confused with `k` as a *width* for 'book').

The default `fontspec <->` in the 5th argument in the `\DeclareFontShape` command means that all sizes are to come from the same font outline (remember if this was a METAFONT font with different design sizes like CM it would be much more complex).

If the face has only a few variants, you can create any other entries for bold, italic, slanted, etc. with the relevant weight and width and shape values pointing at the relevant outline file.

If you want one font to substitute for a missing one (for example italics to substitute for slanted in a typeface which has no slanted variant of its own) give the `ssub` ('silent substitution') command in the `fontspec`: for example to make all references to `sl` (slanted) type use an existing italic font, make the 5th argument like this:

```
{<-> ssub * zor/m/it}
```

If you find the x-height of a font too big or too small to sort well with another font you are using, you can specify an `s` ('scale') factor in this argument instead: this example will shrink the result to 80% of normal:

```
{<-> s * [0.8] zorkly}
```

## 12. Update the index and the map files

Run your T<sub>E</sub>X indexer program (see step 4 in the procedure on p. 82) so that *updmap* can find the files it needs.

```
texhash
```

(or whatever your installation calls it). Then run *updmap* to add the map file:

```
updmap --enable Map=zor.map
```

This updates the maps for *dvips*, *pdfL<sub>A</sub>T<sub>E</sub>X*, and others. Finally, run the T<sub>E</sub>X indexer program again so that it can find the map in its new location.

```
texhash
```

Now you can `\usepackage{foozork}` in your L<sup>A</sup>T<sub>E</sub>X file to make it the default font. To use the font incidentally instead of as the default, you can use the commands you added at the end of step 10 in the procedure on p. 152:

```
This is {\zorkfamily ZORK} or \textzork{ZORK}
```

### 8.3.3 The L<sup>A</sup>T<sub>E</sub>X font catalogue

The L<sup>A</sup>T<sub>E</sub>X Font Catalogue is a web site created and maintained by Palle Jørgensen at <http://www.tug.dk/FontCatalogue/>. It lists over 200 typefaces for use with L<sup>A</sup>T<sub>E</sub>X, many of them available nowhere else, with samples and links to the directories on CTAN where you can download them. You can ~~waste~~ spend many fascinating hours downloading and installing them and trying them out in your documents.

For newcomers, installing a new typeface can appear challenging, when described as I have done for Postscript fonts. But the typefaces in the

LaTeX Font Catalog are prebuilt for LaTeX, so all you have to do is download the .zip file, unzip it into your personal TeX directory, and move the subdirectories into the right places. A worked example is the best way to describe this.

Let's suppose we want to install the sans-serif Kurier typeface. This is nothing to do with the Courier typewriter face, but was designed in pre-computing times by Ma<sup>a</sup>gorzata Budyta, and digitized and extended by Janusz M Nowacki (thanks to the GUST web site<sup>16</sup> for this information).

1. If we click on the name in the sans-serif page of the Catalog<sup>17</sup>, we can see a sample paragraph, and we can click on the link at the bottom of the page<sup>18</sup> to go to the CTAN directory where the typeface is stored.
2. Here there is a brief README file, and links to the individual font subdirectories, but most importantly there is a link at the top of the page to the .zip file containing it all. Download this and unzip it straight into your personal TeX directory (see § 1.2 for what this is and where to create it).
3. Now open your personal TeX directory in your directory browser (Windows: *My Computer* or just *Computer*; Mac: *Finder*; Linux: *nautilus* or *dolphin*). You will see that inside the kurier directory there are subdirectories called doc, fonts, and tex (see Figure 8.1).
4. Drag and drop each of those subdirectories into your Personal TeX directory (texmf). If directories with the same names already exist, your system will probably ask if you want the new ones merged with the existing ones: the answer is yes, so click OK. In some typefaces there may be more subdirectories than shown here for Kurier: do the same with them all.
5. The next step is to find the .map file that LaTeX needs to set up the link between its short (KB) font names and the long ones used by PDF and Postscript output. This will be in the fonts/map/dvips/fontname subdirectory (see Figure 8.2).

In this case there are many, but the one we want is just called kurier.map (the others are there in case you only wanted to install a single font, not the whole typeface).

Carefully note down the directory path to this file (in my case, ~/texmf/fonts/map/dvips/kurier/kurier.map).

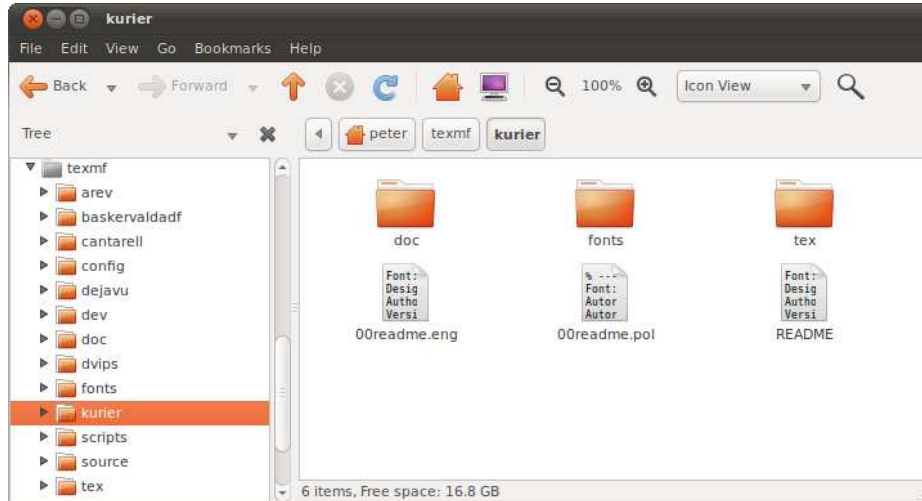
---

<sup>16</sup> <http://www.gust.org.pl/projects/e-foundry/kurier-iwona>

<sup>17</sup> <http://www.tug.dk/FontCatalogue/sansseriffonts.html>

<sup>18</sup> <http://www.ctan.org/tex-archive/fonts/kurier/>

Figure 8.1: Layout of a font zip file downloaded from CTAN



- The last step is to run the font map update program *updmap* to enable use of the map file. You need to do this in a terminal or command window, by typing the command `updmap`. This reloads all your font maps, so it takes a few minutes to run.

Once that's done, you can `\includepackagekurier` in your documents and start using the typeface.

Figure 8.2: Location of the map file for a typeface downloaded from CTAN

