

# The L<sup>A</sup>T<sub>E</sub>X Graphics Companion

Supplementary material

Edited by Michel Goossens (CERN)

*Version February 17, 2008*

*Please send your comments to `michel.goossens@cern.ch`*

Free complement to “The L<sup>A</sup>T<sub>E</sub>X Graphics Companion”, Second Edition.

©Michel Goossens and Sebastian Rahtz (2007)

## Work history

- *August 2007* First general release (for LGC Second Edition)
- *October 2007* Add sections on OpenType features and tables
- *November 2007* Add description of pdftosrc and update information on pdftk to version 1.41
- *November 2007* Separate X<sub>Y</sub>T<sub>E</sub>X material into separate chapter and vastly expand it
- *January 2008* Limit to updated LGC material. Move X<sub>Y</sub>T<sub>E</sub>X into separate document
- *February 2008* Added material about PDF 1.7 and the overprint package (suggestions by Damien Wyart). Use hyperref package to allow easier access to various sections (non-trivial exercise since many T<sub>E</sub>X commands were redefined for the printed version of the book).

# Contents

List of Figures	v
List of Tables	vii
Preface	ix
<b>21 PostScript fonts and beyond</b>	<b>1</b>
21.1 Font formats: an overview	2
21.1.1 A brief history	2
21.1.2 PostScript Type 1 and TrueType: two different approaches	4
21.1.3 Unicode: the universal character encoding	5
21.1.4 OpenType	5
21.2 Typography: combining characters for optimal readability	18
21.2.1 The font metric files	18
21.2.2 T <sub>E</sub> X virtual fonts	23
21.2.3 T <sub>E</sub> X font encodings	26
21.2.4 PostScript font encodings	28
21.2.5 Types of T <sub>E</sub> X fonts	30
21.2.6 Types of PostScript fonts	30
21.2.7 Making all those files work together	32
21.3 PSNFSS: using freely available PostScript Type 1 fonts	35
21.3.1 The standard PSNFSS system	36
21.3.2 pifont—Accessing Pi and Symbol fonts	38
21.3.3 Installing Charter and Utopia	41
21.4 Using commercial PostScript Type 1 fonts with L <sup>A</sup> T <sub>E</sub> X	44
21.4.1 Installing Optima	44
21.5 Using TrueType fonts with pdfT <sub>E</sub> X	46
21.5.1 A predefined setup for using Microsoft Windows TrueType fonts	46
21.5.2 Doing it yourself: under the hood	49
21.5.3 Unicode support with <i>Cyberbit</i>	51
21.6 Installing OpenType Fonts in L <sup>A</sup> T <sub>E</sub> X	53
21.6.1 Creating the L <sup>A</sup> T <sub>E</sub> X font instances	55

21.6.2	Using the LCDF Typetools . . . . .	57
21.6.3	Using the Minion Pro OpenType font . . . . .	58
21.7	Classifying PostScript fonts . . . . .	67
21.8	Font encoding tables . . . . .	74
<b>22</b>	<b>PostScript and PDF tools</b>	<b>87</b>
22.1	Display languages: PostScript, PDF, and SVG . . . . .	87
22.1.1	The PostScript language . . . . .	88
22.1.2	PDF: the Portable Document Format . . . . .	90
22.1.3	SVG for Scalable Vector Graphics . . . . .	91
22.1.4	Comparing an example of PostScript, PDF, and SVG . . . . .	92
22.2	DVI to PostScript drivers and dvips . . . . .	97
22.2.1	The dvips PostScript driver . . . . .	98
22.2.2	Command line and configuration file options . . . . .	98
22.2.3	Paper sizes . . . . .	105
22.2.4	Interaction with PostScript . . . . .	107
22.2.5	Font support . . . . .	109
22.2.6	Special hooks . . . . .	111
22.2.7	Debugging . . . . .	116
22.3	Ghostscript, a PostScript interpreter . . . . .	116
22.3.1	Ghostscript options and initialization . . . . .	118
22.3.2	Ghostscript and fonts . . . . .	122
22.3.3	Selecting an output device . . . . .	122
22.3.4	Interactive Ghostscript versions . . . . .	129
22.3.5	Ghostscript applications . . . . .	129
22.4	PostScript page-manipulation tools . . . . .	135
22.4.1	The psutils suite . . . . .	137
22.4.2	Adding labels to included pictures with psfrag . . . . .	145
22.5	Producing PDF from various sources . . . . .	146
22.5.1	The programs dvipdfm and dvipdfmx . . . . .	147
22.5.2	From PostScript to PDF with pst-pdf . . . . .	149
22.5.3	Generating PDF from L <sup>A</sup> T <sub>E</sub> X . . . . .	151
22.6	PDF manipulation tools . . . . .	154
22.6.1	pdftk . . . . .	154
22.6.2	mbtPDFasm . . . . .	162
22.6.3	Using java for handling PDF files . . . . .	165
22.6.4	Handling PDF document with L <sup>A</sup> T <sub>E</sub> X . . . . .	169
22.6.5	Flipping PDF pages . . . . .	175
22.6.6	The Glyph and Cog tools . . . . .	175
22.7	Color in the printing industry and separation . . . . .	181
22.7.1	Color separation . . . . .	182
22.7.2	Color separation using L <sup>A</sup> T <sub>E</sub> X and dvips . . . . .	184
22.7.3	Overprinting . . . . .	189
	<b>Index of Commands and Concepts</b>	<b>195</b>
	<b>People</b>	<b>201</b>

# List of Figures

21.1	Using OpenType's advanced typographic features in Adobe InDesign . . . . .	12
21.2	Opentype Unicode support in OpenOffice . . . . .	13
21.3	Microsoft's <i>Fonts Extension</i> . . . . .	14
21.4	Font bounding boxes at work . . . . .	20
21.5	Putting bounding boxes on a line . . . . .	21
21.6	Two common ligatures . . . . .	21
21.7	The mechanics of kerning . . . . .	22
21.8	The T1 font layout ( <i>Times Roman</i> ). . . . .	27
21.9	The TeXBase1 font layout ( <i>Times Roman</i> ). . . . .	28
21.10	Multiple Master typeface design space. . . . .	33
21.11	The two-axis Multiple Master Myriad sans serif font . . . . .	33
21.12	The three-axis Multiple Master Minion serif font . . . . .	33
21.13	Files and processes used by $\text{\TeX}$ , a dvi-to-PostScript driver and PostScript. . . . .	35
21.14	Adobe Utopia font sample . . . . .	42
21.15	Bitstream Charter font sample . . . . .	43
21.16	Text sample of Adobe Optima and Bitstream Meridien and Univers . . . . .	45
21.17	Weights and widths available for the Bitstream Univers font . . . . .	46
21.18	Example of using TrueType fonts with pdf $\text{\LaTeX}$ . . . . .	48
21.19	TrueType font sample of Microsoft Times New Roman typeset with pdf $\text{\TeX}$ . . . . .	51
21.20	Multi-lingual Unicode document typeset with the Cyberbit TrueType font . . . . .	54
21.21	Types of figures with Minion Pro . . . . .	64
21.22	Multilingual math typesetting with Minion Pro . . . . .	68
22.1	Stamping pages using dvips . . . . .	112
22.2	Mirror-image printing with dvips. . . . .	114
22.3	Example of the use of ghostview. . . . .	130
22.4	Example of the use of evince. . . . .	130
22.5	Multiple logical pages on one physical sheet, using multi.pro . . . . .	136
22.6	Nine logical pages on one output page . . . . .	141
22.7	Various ways to generate PDF from $\text{\TeX}$ . . . . .	155
22.8	Hypertext document generated with pdf $\text{\LaTeX}$ . . . . .	156

22.9	Example of the use of xpdf. . . . .	176
22.10	The separation of colors in the CMYK model. . . . .	183
22.11	Simple PSTricks color example . . . . .	184
22.12	Example of simple color separations. . . . .	188
22.13	Color separation of a bitmap image using aurora. . . . .	188
22.14	Detail of color separation of a bitmap image. . . . .	188

# List of Tables

21.1	Fonts used by PSNFSS packages . . . . .	36
21.2	Sample texts for standard PostScript fonts. . . . .	37
21.3	The characters in the PostScript font ZapfDingbats . . . . .	39
21.4	Glyphs in the PostScript font Symbol . . . . .	39
21.5	List of Walter Schmidt's PostScript Type 1 support packages . . . . .	44
21.6	Minion Pro Regular font layout . . . . .	60
21.7	The Pi in the PostScript font Minion Pro . . . . .	65
21.8	Encoding support by language in the MinionPro package . . . . .	66
21.9	Font selections available for various encodings in the MinionPro package . . . . .	66
21.10	The standard PostScript fonts in the <i>Fontname</i> scheme. . . . .	67
21.11	Font suppliers defined in the <i>Fontname</i> scheme. . . . .	69
21.12	<i>Fontname</i> weight codes. . . . .	70
21.13	<i>Fontname</i> width codes. . . . .	70
21.14	Font encoding table, by name. . . . .	74
21.15	Font encoding table, by number. . . . .	81
22.1	Standard US, ISO, and Japanese paper sizes . . . . .	124
22.2	Tools in the psutils set . . . . .	138





# Preface

This booklet is a free complement to the second edition of the *ËTËX Graphics Companion*. The material substantially extends what was covered in Chapters 10 and 11 of the first edition. It includes a discussion of DVI-to-PostScript drivers, the free program Ghostscript to view PostScript and PDF files, tools for manipulating PostScript and PDF files, and how to combine the latest font technologies (PostScript Type 1 and OpenType) with ËTËX.

## PostScript and PDF

Why do we want to talk about PostScript and PDF so much? PostScript has been established for more than two decades as an extremely flexible page-description language and it remains the tool of choice for professional typesetters. Among the features that make it so attractive are:

- the quantity, quality, and flexibility of Type 1 fonts;
- the device-independence and portability of files;
- the quality of graphics and the quantity of drawing packages to generate it;
- the facilities for manipulating text;
- the mature color-printing technology;
- the encapsulation conventions that make it easy to embed PostScript graphics;
- the availability of screen-based implementations (e.g., Ghostscript/GhostView).

PostScript has spawned an enterprising child, the PDF (*Portable Document Format*) language, used by Adobe Acrobat and now well established as an exchange format for documents on the Web. Designed for screen display with hypertext features, PDF offers a new degree of portability and efficiency. ËTËX can also produce “rich” PDF documents, and versions of TËX (e.g., pdf<sub>l</sub>atex) that produce PDF directly are available.

Please note that the absence of a given package or tool in this booklet in no way implies that we consider it as less useful or of inferior quality. Moreover, as this document is continuously updated, please feel free to point us to a new interesting developments, so that we can consider mentioning them in a next version of the text.

This complement is work in progress and consists of three chapters. The first two are an updated and expanded version of Chapters 10 and 11 of the *ƉTƉX Graphics Companion* (1st Edition).

**PostScript fonts and beyond** describes the ins and outs of using PostScript fonts with ƉTƉX. It also looks at the latest developments on how to integrate TrueType and OpenType fonts by creating TƉX-specific auxiliary files (TƉX metrics, virtual fonts, etc.).

**PostScript and PDF tools** describes some freely available programs, in particular dvips and pdfflatex to generate PostScript and PDF, ghostscript and ghostview to manipulate and view PostScript and PDF, plus a set of other tools that facilitate handling PostScript and PDF files and conversions.

Comments are welcome and can be addressed to `michel.goossens@cern.ch`.

Michel Goossens  
January 2008

# PostScript fonts and beyond

In this chapter we look at the most basic type of graphical object in documents: the characters that form the words. Character shapes (“glyphs”) are not a direct part of the  $\text{\TeX}$  system; all  $\text{\TeX}$  wants to know about them is some metric information, such as their width or height. It is the task of the post-processing stage (the backend of  $\text{\pdfTeX}$  or a device driver, such as  $\text{\dvips}$  which reads the  $\text{\.dvi}$  file as output by  $\text{\TeX}$ ) to produce the actual graphical representation of the page. For this stage information about the actual shapes of the characters is needed and this information is stored in so-called fonts (collections of characters) for which many different storage formats exist. Thus in principle any existing font can be used with  $\text{\TeX}$  provided that the metric information  $\text{\TeX}$  needs is available or can be generated and that a procedure exists that understands the format in which the fonts are stored and can insert it into the output file.

Donald Knuth developed a companion program to  $\text{\TeX}$ , *MetaFont*, for generating fonts to be used with  $\text{\TeX}$  (Chapter 3 of *κ e LaTeX Graphics Companion* looked briefly at *MetaFont*’s drawing capabilities). For quite some time only fonts designed with *MetaFont* were available to  $\text{\TeX}$  users, with the result that  $\text{\TeX}$  or  $\text{\LaTeX}$  documents had an easily identified look and feel—mainly a result of the use of the Computer Modern fonts. Given that the  $\text{\TeX}$  community is very small compared to that of other typesetting systems very few font designers have produced fonts in *MetaFont*. Therefore, access for  $\text{\TeX}$  engines to the literally thousands of fonts available commercially in other formats, in particular PostScript, TrueType, and, more recently, OpenType, has become a *must*.

Although at the beginning it was quite difficult to integrate PostScript fonts into  $\text{\LaTeX}$  packages, the release of  $\text{\LaTeX 2}_{\epsilon}$  and its new font selection scheme (NFSS, see Chapter 7 of [15]) made accessing the large set of PostScript fonts more straightforward. Nowadays, documents routinely combine  $\text{\TeX}$ ’s superior typesetting quality with all the professionally designed typefaces produced, mainly in PostScript, but also in TrueType and OpenType. The current chapter will introduce you to solutions to achieve this in a convenient way.

After a historic overview of modern font technologies, including a brief description of their respective technical capabilities, we take a closer look at the basic issues concerned with typesetting and how  $\text{\TeX}$  and PostScript, working together, address this problem (how metric information is handled, the different types of  $\text{\TeX}$  and PostScript fonts, how they are encoded, i.e., how one can access individual characters of a font, etc.) We then explain how you can use the “basic” PostScript fonts, as they are defined in the PSNFSS system (a collection of small packages and accompanying files for  $\text{\LaTeX}$ ), which makes it easy to use a large number of common PostScript fonts out of the box) and how to easily down-

load and install a few instances of freely available fonts. We extend the discussion to where to download and install the  $\text{\TeX}$  support files for commercially available fonts that you might have bought. Since many  $\text{\TeX}$  users have de facto access to a lot of TrueType fonts that come with their operating system, we devote the next section to the use of TrueType fonts with  $\text{\pdf}$ , in particular how one can use a large Unicode TrueType font for typesetting in many different scripts and languages. We are then ready to discuss a few recent  $\text{\TeX}$  packages which take advantage of the enriched possibilities of the OpenType technology. We end the chapter with a discussion of *Fontname*, also known as the “Berry” font naming scheme, which is important to uniquely identify and handle all  $\text{\TeX}$  support files of the large number of fonts that are available on current operating systems.

## 21.1 Font formats: an overview

The current main font formats are PostScript Type 1 (Type 1), TrueType (TT), and OpenType (OT), an integrated superset of the first two. All three are based on font outline technologies, are multi-platform, and have their technical specifications openly available. These formats can be run on any recent computer platform and their character outlines (“glyphs”) are described mathematically as functions operating on points, lines and curves. The character representations are resolution independent and can be scaled to any size. These technologies implement “hinting” by associating additional information with each character to help the rasterization engine optimize their representation on any given output device.

### 21.1.1 A brief history

#### 21.1.1.1 Adobe and its PostScript Type 1

When Adobe launched PostScript in 1984, it supported two different types of fonts formats: Type 1,<sup>1</sup> the more sophisticated one with support for hinting and data compression, and Type 3, a more general (almost all PostScript graphics operators are allowed) but less optimized variant. At first Adobe did not publish the specification of its PostScript Type 1 format (the Type 3 spec was public), which helped Adobe take a large part of the commercial typography market but upset the other font foundries.

Apple, which also was founded in the early nineteen eighties, adopted PostScript as page description language for its Apple LaserWriter printer in 1985. Soon also other high-end image setting machines adopted PostScript as their native language. At about the same time the introduction of affordable desktop publishing software, such as Pagemaker, Freehand, set off a revolution in page layout technology, and PostScript backends appeared for most graphics programs, thus adding to the potential market for professional PostScript Type 1 fonts. Because of its reliability, its wide selection of fonts available, its clever rasterizing engine and superior hinting mechanism, historically PostScript has been the preferred font format of professional designers, publishers and printshops.

Concurrently Adobe had developed an “interactive” version of PostScript, called *Display PostScript*, that ran (somewhat slowly) on personal computers to allow displaying PostScript data on-screen. Although some computer manufacturers agreed to take out (and pay) software licences, Apple and Microsoft were quite unwilling to pay the royalties requested by Adobe and, moreover, to hand control to Adobe over a vital part of their operating system.

In the first part of the 1990s Adobe also developed the PostScript Type 1 multiple master (MM) format as an extension of PostScript Type 1. Essentially, it allows two (or more) design variations to be encoded on a given design axis (such as weight, width, optical size). Afterwards, any in-between state (*instance*) may be generated by the user as required.<sup>2</sup>

<sup>1</sup>See [http://partners.adobe.com/public/developer/en/font/T1\\_SPEC.PDF](http://partners.adobe.com/public/developer/en/font/T1_SPEC.PDF).

<sup>2</sup>The technology never really took off and since 2000 Adobe has abandoned developing multiple master fonts since most

### 21.1.1.2 TrueType fonts

The major system software vendors (Apple, Microsoft, IBM) had been thinking about scaleable font technology support at the level of their respective operating systems since they realized that it would guarantee much better screen display, compared to pre-generated bitmaps which only look good at their design sizes, and unacceptably jagged at all others. For instance in the late 1980s Apple had developed an in-house scaleable font technology, *Royal*, later renamed to TrueType.<sup>1</sup> The TrueType specification was public and already in 1991 native TrueType support appeared in Apple's Mac System 7 and Microsoft's Windows 3.1.

TrueType fonts use a different outline model from PostScript, and also the approach to hinting is different. The font instances contain both screen and printer font data in a single component. This makes the fonts easy to install. Although TrueType fonts support Unicode and can theoretically contain over 65.000 characters, they rarely feature more than some 220 characters. Moreover, TrueType font formats are platform-dependent.

### 21.1.1.3 Two competing technologies

Adobe reacted to the advent of TrueType by publishing in 1990 the PostScript Type 1 font format specification [1]. A few years later, it introduced the *Adobe Type Manager* (ATM) software, which scales PostScript Type 1 fonts for screen display, and supports imaging on non-PostScript printers.

Thus by the end of the 1990s there were two widely-used outline font specifications, TrueType, built into the operating systems used by most desktop computers, and PostScript Type 1, the de facto standard for the graphic arts and the publishing industry. Moreover, as time went by, the practical differences had begun to blur. On the one hand, support for TrueType became standard in PostScript 3, while on the other hand, besides native TrueType support, PostScript Type 1 rasterizing technology was incorporated into Windows 2000, Windows XP, and Mac OS X.

### 21.1.1.4 The best of two worlds: OpenType

The OpenType<sup>2</sup> font format was jointly developed by Adobe and Microsoft to combine the best features of the TrueType and PostScript Type 1 technologies. It was first presented in 1996 and its use and support has been steadily increasing since about 2000.

OpenType fonts contain both the screen and printer font data in a single component. The OpenType format can contain either TrueType or PostScript font data. It supports expanded character sets (up to 65.000) and special typographic features. These may include various versions of figures (tabular, old-style, lining), small caps, ligatures, ordinals, and other extras. While OpenType allows type designers to build complex fonts, not many fonts take advantage of these possibilities. Most OpenType fonts available today are simply converted PostScript fonts, limited to 220 characters in a set.

OpenType fonts are platform independent and can thus be used on all operating systems.

---

applications cannot handle them and for a large majority of users it often makes more economic sense to buy a fontset as multiple separate fonts. Adobe now concentrates on releasing OpenType fonts to replace their multiple master equivalents (e.g., the Minion and Myriad typefaces).

<sup>1</sup>See e.g., <http://developer.apple.com/fonts/>, and <http://www.microsoft.com/typography>.

<sup>2</sup>See Adobe's Web pages <http://store.adobe.com/type/opentype/main.html>, and <http://blogs.adobe.com/typblography/TT%20PS%20OpenType.pdf>, or Microsoft's Web page <http://www.microsoft.com/typography/OTSPEC/default.htm>.

### 21.1.2 PostScript Type 1 and TrueType: two different approaches

TrueType and PostScript Type 1 fonts use different mathematical representations to describe the curves defining the font outlines.<sup>1</sup> OpenType, being a superset, can have either kind of outlines.

TrueType describes its curves by quadratic B-splines, while PostScript Type 1 uses cubic Bézier curves. This means, in practice, that the shapes of real-world fonts tend to take more points in TrueType, even though the kind of mathematics used to describe the curves is simpler. Any quadratic spline can be converted to a cubic spline with essentially no loss. A cubic spline can be converted to a quadratic with arbitrary precision, but there will be a slight loss of accuracy in most cases. Thus it is easy to convert TrueType outlines to PostScript Type 1 outlines (the “Type 42” PostScript font format is a PostScript wrapper around a TrueType font for use in PostScript interpreters), harder to do the reverse.

The approach to hinting is different in both technologies. PostScript Type 1<sup>2</sup> takes a *declarative* approach and lets a *smart* PostScript interpreter do the work. It tells the rasterizer what features ought to be controlled, and the rasterizer interprets these using its own “intelligence” to decide how to do it. Therefore, when the PostScript interpreter is upgraded, the rasterization can be improved.

On paper, the hinting potential of TrueType<sup>3</sup> should be superior to that of PostScript Type 1 fonts, since TrueType hints can do all that PostScript Type 1 can, and more. Indeed TrueType takes an *algorithmic* or programming approach and uses the very flexible and complete instructions set of the TrueType language. Thus TrueType puts all the hinting information into the font to control exactly how it will appear when rasterized. TrueType interpreters can be quite “dumb” and limit themselves to simply execute what they have been “instructed” to do. Thus, although a TrueType font developer can finetune what happens when a font is rasterized under different conditions, it requires serious effort, expertise, and high-end tools to actually take advantage of this greater hinting potential. As a result, high-quality TrueType fonts, which exploit the true potentials of TrueType hinting only quite rare. Moreover, when using complex hinting the introduction of a new rasterizer might require major changes to the TrueType code in order to be able to optimally display existing fonts.

PostScript Type 1 needs two separate files for its font data: one for the character outlines (`.pfb`), and the other for the metrics data (`.afm` on Linux, `.pfm` on Windows), containing character widths, kerning pairs, and a description of how to construct composites. TrueType fonts have all the data in a single file. Nevertheless this single TrueType font file is often twice larger than the two PostScript Type 1 files combined due to the presence in the TrueType fonts of extensive “hinting” instructions.

Generally speaking, PostScript Type 1 fonts have some advantages simply from being the longer-established standard, especially for serious graphic arts work. Service bureaus are standardized on, and have large investments in, PostScript Type 1 fonts. Most of the fonts which have “expert sets” of old style figures, extra ligatures, true small capitals and the like are in that format.

#### 21.1.2.1 Interoperability

In principle one can mix TrueType and PostScript Type 1 fonts with the caveat that the TrueType and PostScript Type 1 instances of the fonts may not have exactly the same names on the given operating system. Indeed, the fact that fonts exist with identical menu names or PostScript Type 1 font names confuses the operating system or the application programs, with often unpredictable results.

Also, if using Windows, one may find that metrically-similar PostScript Type 1 fonts get substituted for the Windows TrueType system fonts at output time: *Times New Roman* becomes *Times Roman*, and *Arial* becomes *Helvetica*. Although the basic spacing of the substituted fonts is identical, their kerning pairs are not. This can cause text to reflow if one switches between two “almost identical” fonts if your

<sup>1</sup>See <http://www.trueType.demon.co.uk/articles/ttvst1.htm>.

<sup>2</sup>See Dadid Lemon’s *Basic Type 1 hinting* (<http://www.pyrus.com/downloads/hinting.pdf>).

<sup>3</sup>See the URL <http://www.microsoft.com/typography/hinting/tutorial.htm>, Vincent Connare’s *Basic hinting philosophies and TrueType instructions*.

typesetting program (e.g.,  $\text{\TeX}$ ) supports kerning pairs. Thus care must be taken to ensure that you use the correct font all through the complete production chain.

### 21.1.3 Unicode: the universal character encoding

Unicode is an international standard<sup>1</sup> for representing characters using a multi-byte platform-independent encoding for covering all the world languages (including some “artificial” ones, such as mathematical symbols and the international phonetic alphabet). Unicode deals with characters rather than glyphs. That is, it only deals with semantic rather than typographic distinctions (with a few exceptions for compatibility with existing standards). Therefore there is no place for glyph variants, such as unusual ligatures, old style numbers, or small caps within Unicode itself; the Unicode standard assumes that such distinctions will be made elsewhere. Therefore, font formats, which supports such distinctions, such as OpenType (see Section 21.1.4), need to be layered on top of Unicode. Alan Wood’s maintains a useful website (<http://www.alanwood.net/unicode/>) which describes numerous resources for Unicode and multilingual support in HTML, fonts, web browsers and other applications.

Most current operating systems (Linux, Mac OS X and Windows XP) have direct support for Unicode at the basic system level. For instance, apart from switching between different language keyboards, these operating systems offer means of directly accessing any Unicode character in any font (e.g., on Mac OS X via the *Character Palette* and on Microsoft Windows 2000/XP via the *Character Map* in *System Tools* in the *Accessories* submenu.)

### 21.1.4 OpenType

The OpenType font format was developed jointly by Microsoft and Adobe as an extension of the TrueType font format. OpenType addresses the following goals:

- supports PostScript Type 1 outlines and hints;
- supports TrueType tables and hints;
- supports advanced typographic features by way of new tables for glyph positioning and substitution;
- supports multiple platforms;
- supports international character sets by using Unicode;
- offers better protection for font data;
- features smaller file sizes to make font distribution more efficient.

Sometimes OpenType fonts are referred to as TrueType Open v.2.0 fonts. PostScript Type 1 data included in OpenType fonts may be directly rasterized or converted to the TrueType outline format for rendering, depending on which rasterizers have been installed in the host operating system. Users do not need to know which outlines are actually present. One can say that OpenType enters TrueType and PostScript Type 1 in a common wrapper. OpenType tables include the current TrueType tables plus some additional tables for advanced typographic features. The representation of PostScript Type 1 font software in an OpenType font uses Adobe’s Compact Font Format (CFF) with Type 2 charstrings, which is a more compact representation of the same information in PostScript Type 1 (a gain of about a factor of two, on average, when no glyphs and features are added).

The OpenType format supports *features* equivalent to most of the advanced features of existing TrueType and PostScript formats, such as Adobe’s CID technology for Asian fonts, and extended mul-

<sup>1</sup>The current version is 5.0 [20] and it has been defined by the members of the Unicode Consortium, which includes major computer corporations, software producers, database vendors, research institutions, international agencies, various user groups, and interested individuals, see <http://www.unicode.org>.



tilingual character sets. However, multiple master fonts are not part of the OpenType specification. OpenType fonts may contain more than 65,000 glyphs, which allows a single font file to contain many nonstandard glyphs, such as old-style figures, true small capitals, fractions, swashes, superiors, inferiors, titling letters, contextual and stylistic alternates, and a full range of ligatures. OpenType fonts thus offers rich linguistic support combined with advanced typographic control. Feature-rich Adobe OpenType fonts are often distinguished by the word “Pro,” being part of the font name. OpenType fonts can be installed and used alongside PostScript Type 1 and TrueType fonts.

OpenType, which is based on Unicode, significantly simplifies font management and the publishing process by ensuring that all of the required glyphs for a document are contained in one cross-platform font file throughout the workflow.

The text model of OpenType is that applications store text using the underlying Unicode characters, and apply formatting to get at the specific desired glyphs. In addition to the Unicode mapping of default glyphs, the font has OpenType layout tables which tell it which glyphs to use when other forms are desired instead, such as small caps or swashes. These tables also specify which glyphs should turn into ligatures, or when a script font needs different glyphs for a letter when it is at the beginning, middle or end of a word, or is a word by itself.

Having the transformations distinct from the underlying text enables table-driven automatic glyph substitution, which does not need to be one for one; one glyph can be substituted for several (such as the “ffi” ligature, which remembers that the underlying text contains the characters “f-f-i” in searching), or multiple glyphs can be substituted for a single one. Glyph substitution can be context sensitive, or it can be activated by explicit user demand. All of is not very essential for Latin-based languages, such as Spanish and English, but it becomes mandatory for proper typesetting of languages that use “complex scripts”, such as Arabic or the Indic languages, since having letters take different forms based on their position in the word is a basic part of how Arabic works.

OpenType layout features can be used to position or substitute glyphs. For any character, there is a default glyph and positioning behavior. The application of layout features to one or more characters may change the positioning, or substitute a different glyph.

There are several advantages of using a large OpenType font over currently available “expert sets” and “alternates”. First, one only has to deal with one font file, rather than being cluttered with a whole set of supplemental fonts. Second, there can be kerning between glyphs that might otherwise have been in separate fonts. Finally, the user can turn on ligatures, smallcaps, or old-style figures, much like bold or italic styling, without switching fonts.

Historically, some of the highest quality typefaces have included different designs for different print sizes. Rather than using its multiple masters technology, most of Adobe’s OpenType fonts now include four optical size variations: caption, regular, subhead and display. Called “Opticals,” these variations have been optimised for use at specific point sizes. Although the exact intended sizes vary by family, the general size ranges include: caption (6–8 point), regular (9–13 point), subhead (14–24 point) and display (25–72 point).

#### 21.1.4.1 OpenType tables

OpenType font files contain tables that contain either TrueType or PostScript outline font data and the data in these tables are used by rendering programs to render the TrueType or PostScript glyphs. Moreover, some of the data is independent of the particular outline format used.<sup>1</sup>

OpenType fonts first contain a number of *required* tables.

<sup>1</sup>The structure of an OpenType font file is described at the URL <http://www.microsoft.com/typography/otspec/otff.htm>; a short description of the contents of the tables is at the URL <http://www.microsoft.com/typography/otspec/recom.htm>.



<b>cmap</b>	Character to glyph mapping	<b>maxp</b>	Maximum profile
<b>head</b>	Font header	<b>name</b>	Naming table
<b>hhea</b>	Horizontal header	<b>OS/2</b>	OS/2 and Windows specific metrics
<b>hmtx</b>	Horizontal metrics	<b>post</b>	PostScript information

For OpenType fonts based on TrueType outlines, the following tables are used:

<b>cvt</b>	Control Value Table	<b>glyf</b>	Glyph data	<b>prep</b>	CVT Program
<b>fpgm</b>	Font program	<b>loca</b>	Index to location		

For OpenType fonts based on PostScript another set of tables containing data specific to PostScript fonts are used instead of the tables listed above:

<b>CFF</b>	PostScript font program (compact font format)
<b>VORG</b>	Vertical Origin

OpenType fonts may contain bitmaps of glyphs, in addition to outlines. Hand-tuned bitmaps are especially useful in OpenType fonts for representing complex glyphs at very small sizes. If a bitmap for a particular size is provided in a font, it will be used by the system instead of the outline when rendering the glyph. For OpenType fonts containing bitmap glyphs three tables are available:

<b>EBDT</b>	Embedded bitmap data
<b>EBLC</b>	Embedded bitmap location data
<b>EBSC</b>	Embedded bitmap scaling data

Finally, advanced typography, vertical typesetting and other special functions are supported with the following tables:

<b>BASE</b>	Baseline data	<b>hdmx</b>	Horizontal device metrics
<b>GDEF</b>	Glyph definition data	<b>kern</b>	Kerning
<b>GPOS</b>	Glyph positioning data	<b>LTSH</b>	Linear threshold data
<b>GSUB</b>	Glyph substitution data	<b>PCLT</b>	PCL 5 data
<b>JSTF</b>	Justification data	<b>VDMX</b>	Vertical device metrics
<b>DSIG</b>	Digital signature	<b>vhea</b>	Vertical Metrics header
<b>gasp</b>	Grid-fitting/Scan-conversion	<b>vmtx</b>	Vertical Metrics

Furthermore, OpenType fonts use a set of script, language and feature tags to structure the information in their tables.

*Script tags* identify the scripts represented in an OpenType font. Each script corresponds to a contiguous character code range in Unicode. Script tags are four-byte character strings composed of up to four letters in the ASCII characters range 0x20–0x7E, padding with blanks (0x20) if required. A list of the most commonly used scripts and their associated tag is given below.

<b>DFLT</b>	Default	<b>cyrl</b>	Cyrillic	<b>hani</b>	CJK Ideographic
<b>arab</b>	Arabic	<b>deva</b>	Devanagari	<b>hebr</b>	Hebrew
<b>armn</b>	Armenian	<b>ethi</b>	Ethiopic	<b>kana</b>	Hiragana
<b>beng</b>	Bengali	<b>geor</b>	Georgian	<b>knda</b>	Kannada
<b>bopo</b>	Bopomofo	<b>grek</b>	Greek	<b>kata</b>	Katakana
<b>brai</b>	Braille	<b>gujr</b>	Gujarati	<b>khmr</b>	Khmer
<b>byzm</b>	Byzantine Music	<b>guru</b>	Gurmukhi	<b>lao</b>	Lao
<b>cans</b>	Canadian Syllabics	<b>jamo</b>	Hangul Jamo	<b>latn</b>	Latin
<b>cher</b>	Cherokee	<b>hang</b>	Hangul	<b>mlym</b>	Malayalam

<b>mong</b>	Mongolian	<b>sinh</b>	Sinhala	<b>thai</b>	Thai
<b>mymr</b>	Myanmar	<b>syrc</b>	Syriac	<b>tibt</b>	Tibetan
<b>ogam</b>	Ogham	<b>taml</b>	Tamil	<b>yi</b>	Yi
<b>orya</b>	Oriya	<b>telu</b>	Telugu		
<b>runr</b>	Runic	<b>thaa</b>	Thaana		

When the table with the list of scripts is searched for a script, and no entry is found, and there exists an entry for the `DFLT` script, then this entry must be used. Furthermore, the default script can only contain a single, default, language.

*Language system tags* identify the language systems supported in an OpenType font. Language tags are four-byte character strings composed of up to four letters in the ASCII characters range `0x20–0x7E`, padding with blanks (`0x20`) if required. A list of languages and their tags follows.

<b>ABA</b>	Abaza	<b>BLN</b>	Balante	<b>DNG</b>	Dangme
<b>ABK</b>	Abkhazian	<b>BLT</b>	Balti	<b>DNK</b>	Dinka
<b>ADY</b>	Adyghe	<b>BMB</b>	Bambara	<b>DUN</b>	Dungan
<b>AFK</b>	Afrikaans	<b>BML</b>	Bamileke	<b>DZN</b>	Dzongkha
<b>AFR</b>	Afar	<b>BRE</b>	Breton	<b>EBI</b>	Ebira
<b>AGW</b>	Agaw	<b>BRH</b>	Brahui	<b>ECR</b>	Eastern Cree
<b>ALT</b>	Altai	<b>BRI</b>	Braj Bhasha	<b>EDO</b>	Edo
<b>AMH</b>	Amharic	<b>BRM</b>	Burmese	<b>EFI</b>	Efik
<b>ARA</b>	Arabic	<b>BSH</b>	Bashkir	<b>ELL</b>	Greek
<b>ARI</b>	Aari	<b>BTI</b>	Beti	<b>ENG</b>	English
<b>ARK</b>	Arakanese	<b>CAT</b>	Catalan	<b>ERZ</b>	Erzya
<b>ASM</b>	Assamese	<b>CEB</b>	Cebuano	<b>ESP</b>	Spanish
<b>ATH</b>	Athapaskan	<b>CHE</b>	Chechen	<b>ETI</b>	Estonian
<b>AVR</b>	Avar	<b>CHG</b>	Chaha Gurage	<b>EUQ</b>	Basque
<b>AWA</b>	Awadhi	<b>CHH</b>	Chattisgarhi	<b>EVK</b>	Evenki
<b>AYM</b>	Aymara	<b>CHI</b>	Chichewa	<b>EVN</b>	Even
<b>AZE</b>	Azeri	<b>CHK</b>	Chukchi	<b>EWE</b>	Ewe
<b>BAD</b>	Badaga	<b>CHP</b>	Chipewyan	<b>FAN</b>	French Antillean
<b>BAG</b>	Baghelkhandi	<b>CHR</b>	Cherokee	<b>FAR</b>	Farsi
<b>BAL</b>	Balkar	<b>CHU</b>	Chuvash	<b>FIN</b>	Finnish
<b>BAU</b>	Baule	<b>CMR</b>	Comorian	<b>FJI</b>	Fijian
<b>BBR</b>	Berber	<b>COP</b>	Coptic	<b>FLE</b>	Flemish
<b>BCH</b>	Bench	<b>CRE</b>	Cree	<b>FNE</b>	Forest Nenets
<b>BCR</b>	Bible Cree	<b>CRR</b>	Carrier	<b>FON</b>	Fon
<b>BEL</b>	Belarussian	<b>CRT</b>	Crimean Tatar	<b>FOS</b>	Faroese
<b>BEM</b>	Bemba	<b>CSL</b>	Church Slavonic	<b>FRA</b>	French (Standard)
<b>BEN</b>	Bengali	<b>CSY</b>	Czech	<b>FRI</b>	Frisian
<b>BGR</b>	Bulgarian	<b>DAN</b>	Danish	<b>FRL</b>	Friulian
<b>BHI</b>	Bhili	<b>DAR</b>	Dargwa	<b>FTA</b>	Futa
<b>BHO</b>	Bhojpuri	<b>DCR</b>	Woods Cree	<b>FUL</b>	Fulani
<b>BIK</b>	Bikol	<b>DEU</b>	German (Standard)	<b>GAD</b>	Ga
<b>BIL</b>	Bilen	<b>DGR</b>	Dogri	<b>GAE</b>	Gaelic
<b>BKF</b>	Blackfoot	<b>DHV</b>	Dhivehi	<b>GAG</b>	Gagauz
<b>BLI</b>	Balochi	<b>DJR</b>	Djerma	<b>GAL</b>	Galician

<b>GAR</b>	Garshuni	<b>KAR</b>	Karachay	<b>LAK</b>	Lak
<b>GAW</b>	Garhwali	<b>KAT</b>	Georgian	<b>LAM</b>	Lambani
<b>GEZ</b>	Ge'ez	<b>KAZ</b>	Kazakh	<b>LAO</b>	Lao
<b>GIL</b>	Gilyak	<b>KEB</b>	Kebena	<b>LAT</b>	Latin
<b>GMZ</b>	Gumuz	<b>KGE</b>	Khutsuri Georgian	<b>LAZ</b>	Laz
<b>GON</b>	Gondi	<b>KHA</b>	Khakass	<b>LCR</b>	L-Cree
<b>GRN</b>	Greenlandic	<b>KHK</b>	Khanty-Kazim	<b>LDK</b>	Ladakhi
<b>GRO</b>	Garo	<b>KHM</b>	Khmer	<b>LEZ</b>	Lezgi
<b>GUA</b>	Guarani	<b>KHS</b>	Khanty-Shurishkar	<b>LIN</b>	Lingala
<b>GUJ</b>	Gujarati	<b>KHV</b>	Khanty-Vakhi	<b>LMA</b>	Low Mari
<b>HAI</b>	Haitian	<b>KHW</b>	Khovar	<b>LMB</b>	Limbu
<b>HAL</b>	Halam	<b>KIK</b>	Kikuyu	<b>LMW</b>	Lomwe
<b>HAR</b>	Harauti	<b>KIR</b>	Kirghiz	<b>LSB</b>	Lower Sorbian
<b>HAU</b>	Hausa	<b>KIS</b>	Kisii	<b>LSM</b>	Lule Sami
<b>HAW</b>	Hawaiin	<b>KKN</b>	Kokni	<b>LTH</b>	Lithuanian
<b>HBN</b>	Hammer-Banna	<b>KLM</b>	Kalmyk	<b>LUB</b>	Luba
<b>HIL</b>	Hiligaynon	<b>KMB</b>	Kamba	<b>LUG</b>	Luganda
<b>HIN</b>	Hindi	<b>KMN</b>	Kumaoni	<b>LUH</b>	Luhya
<b>HMA</b>	High Mari	<b>KMO</b>	Komo	<b>LUO</b>	Luo
<b>HND</b>	Hindko	<b>KMS</b>	Komso	<b>LVI</b>	Latvian
<b>HO</b>	Ho	<b>KNR</b>	Kanuri	<b>MAJ</b>	Majang
<b>HRI</b>	Harari	<b>KOD</b>	Kodagu	<b>MAK</b>	Makua
<b>HRV</b>	Croatian	<b>KOK</b>	Konkani	<b>MAL</b>	Malayalam Traditional
<b>HUN</b>	Hungarian	<b>KON</b>	Kikongo	<b>MAN</b>	Mansi
<b>HYE</b>	Armenian	<b>KOP</b>	Komi-Permyak	<b>MAR</b>	Marathi
<b>IBO</b>	Igbo	<b>KOR</b>	Korean	<b>MAW</b>	Marwari
<b>IJO</b>	Ijo	<b>KOZ</b>	Komi-Zyrian	<b>MBN</b>	Mbundu
<b>ILO</b>	Ilokano	<b>KPL</b>	Kpelle	<b>MCH</b>	Manchu
<b>IND</b>	Indonesian	<b>KRI</b>	Krio	<b>MCR</b>	Moose Cree
<b>ING</b>	Ingush	<b>KRK</b>	Karakalpak	<b>MDE</b>	Mende
<b>INU</b>	Inuktitut	<b>KRL</b>	Karelian	<b>MEN</b>	Me'en
<b>IRI</b>	Irish	<b>KRM</b>	Karaim	<b>MIZ</b>	Mizo
<b>IRT</b>	Irish Traditional	<b>KRN</b>	Karen	<b>MKD</b>	Macedonian
<b>ISL</b>	Icelandic	<b>KRT</b>	Koorete	<b>MLE</b>	Male
<b>ISM</b>	Inari Sami	<b>KSH</b>	Kashmiri	<b>MLG</b>	Malagasy
<b>ITA</b>	Italian	<b>KSI</b>	Khasi	<b>MLN</b>	Malinke
<b>IWR</b>	Hebrew	<b>KSM</b>	Kildin Sami	<b>MLR</b>	Malayalam Reformed
<b>JAN</b>	Japanese	<b>KUI</b>	Kui	<b>MLY</b>	Malay
<b>JAV</b>	Javanese	<b>KUL</b>	Kulvi	<b>MND</b>	Mandinka
<b>JII</b>	Yiddish	<b>KUM</b>	Kumyk	<b>MNG</b>	Mongolian
<b>JUD</b>	Judezmo	<b>KUR</b>	Kurdish	<b>MNI</b>	Manipuri
<b>JUL</b>	Jula	<b>KUU</b>	Kurukh	<b>MNK</b>	Maninka
<b>KAB</b>	Kabardian	<b>KUY</b>	Kuy	<b>MNX</b>	Manx Gaelic
<b>KAC</b>	Kachchi	<b>KYK</b>	Koryak	<b>MOK</b>	Moksha
<b>KAL</b>	Kalenjin	<b>LAD</b>	Ladin	<b>MOL</b>	Moldavian
<b>KAN</b>	Kannada	<b>LAH</b>	Lahuli	<b>MON</b>	Mon

<b>MOR</b>	Moroccan	<b>ROM</b>	Romanian	<b>TCR</b>	TH-Cree
<b>MRI</b>	Maori	<b>ROY</b>	Romany	<b>TEL</b>	Telugu
<b>MTH</b>	Maithili	<b>RSY</b>	Rusyn	<b>TGN</b>	Tongan
<b>MTS</b>	Maltese	<b>RUA</b>	Ruanda	<b>TGR</b>	Tigre
<b>MUN</b>	Mundari	<b>RUS</b>	Russian	<b>TGY</b>	Tigrinya
<b>NAG</b>	Naga-Assamese	<b>SAD</b>	Sadri	<b>THA</b>	Thai
<b>NAN</b>	Nanai	<b>SAN</b>	Sanskrit	<b>THT</b>	Tahitian
<b>NAS</b>	Naskapi	<b>SAT</b>	Santali	<b>TIB</b>	Tibetan
<b>NCR</b>	N-Cree	<b>SAY</b>	Sayisi	<b>TKM</b>	Turkmen
<b>NDB</b>	Ndebele	<b>SEK</b>	Sekota	<b>TMN</b>	Temne
<b>NDG</b>	Ndonga	<b>SEL</b>	Selkup	<b>TNA</b>	Tswana
<b>NEP</b>	Nepali	<b>SGO</b>	Sango	<b>TNE</b>	Tundra Nenets
<b>NEW</b>	Newari	<b>SHN</b>	Shan	<b>TNG</b>	Tonga
<b>NHC</b>	Norway House Cree	<b>SIB</b>	Sibe	<b>TOD</b>	Todo
<b>NIS</b>	Nisi	<b>SID</b>	Sidamo	<b>TRK</b>	Turkish
<b>NIU</b>	Niuean	<b>SIG</b>	Silte Gurage	<b>TSG</b>	Tsonga
<b>NKL</b>	Nkole	<b>SKS</b>	Skolt Sami	<b>TUA</b>	Turoyo Aramaic
<b>NLD</b>	Dutch	<b>SKY</b>	Slovak	<b>TUL</b>	Tulu
<b>NOG</b>	Nogai	<b>SLA</b>	Slavey	<b>TUV</b>	Tuvin
<b>NOR</b>	Norwegian	<b>SLV</b>	Slovenian	<b>TWI</b>	Twi
<b>NSM</b>	Northern Sami	<b>SML</b>	Somali	<b>UDM</b>	Udmurt
<b>NTA</b>	Northern Tai	<b>SMO</b>	Samoaan	<b>UKR</b>	Ukrainian
<b>NTO</b>	Esperanto	<b>SNA</b>	Sena	<b>URD</b>	Urdu
<b>NYN</b>	Nynorsk	<b>SND</b>	Sindhi	<b>USB</b>	Upper Sorbian
<b>OCR</b>	Oji-Cree	<b>SNH</b>	Sinhalese	<b>UYG</b>	Uyghur
<b>OJB</b>	Ojibway	<b>SNK</b>	Soninke	<b>UZB</b>	Uzbek
<b>ORI</b>	Oriya	<b>SOG</b>	Sodo Gurage	<b>VEN</b>	Venda
<b>ORO</b>	Oromo	<b>SOT</b>	Sotho	<b>VIT</b>	Vietnamese
<b>OSS</b>	Ossetian	<b>SQI</b>	Albanian	<b>WAG</b>	Wagdi
<b>PAA</b>	Palestinian Aramaic	<b>SRB</b>	Serbian	<b>WA</b>	Wa
<b>PAL</b>	Pali	<b>SRK</b>	Saraiki	<b>WCR</b>	West-Cree
<b>PAN</b>	Punjabi	<b>SRR</b>	Serer	<b>WEL</b>	Welsh
<b>PAP</b>	Palpa	<b>SSL</b>	South Slavey	<b>WLF</b>	Wolof
<b>PAS</b>	Pashto	<b>SSM</b>	Southern Sami	<b>XHS</b>	Xhosa
<b>PGR</b>	Polytonic Greek	<b>SUR</b>	Suri	<b>YAK</b>	Yakut
<b>PIL</b>	Pilipino	<b>SVA</b>	Svan	<b>YBA</b>	Yoruba
<b>PLG</b>	Palaung	<b>SVE</b>	Swedish	<b>YCR</b>	Y-Cree
<b>PLK</b>	Polish	<b>SWA</b>	Swadaya Aramaic	<b>YIC</b>	Yi Classic
<b>PRO</b>	Provencal	<b>SWK</b>	Swahili	<b>YIM</b>	Yi Modern
<b>PTG</b>	Portuguese	<b>SWZ</b>	Swazi	<b>ZHP</b>	Chinese Phonetic
<b>QIN</b>	Chin	<b>SXT</b>	Sutu	<b>ZHS</b>	Chinese Simplified
<b>RAJ</b>	Rajasthani	<b>SYR</b>	Syriac	<b>ZHT</b>	Chinese Traditional
<b>RBU</b>	Russian Buriat	<b>TAB</b>	Tabasaran	<b>ZND</b>	Zande
<b>RCR</b>	R-Cree	<b>TAJ</b>	Tajiki	<b>ZUL</b>	Zulu
<b>RIA</b>	Riang	<b>TAM</b>	Tamil		
<b>RMS</b>	Rhaeto-Romanic	<b>TAT</b>	Tatar		

### 21.1.4.2 OpenType features

*Features* provide information about how to use the glyphs in an OpenType or TrueType font to render a script or language. For example, an Arabic font might have a feature for substituting initial glyph forms, and a Kanji font might have a feature for positioning glyphs vertically. All OpenType Layout features define data for glyph substitution, glyph positioning, or both.

Each OpenType Layout feature has a feature tag that identifies its typographic function and effects. By examining a feature's tag, a text-processing client can determine what a feature does and decide whether to implement it. All tags are four-byte character strings composed of a limited set of ASCII characters (range 0x20–0x7E).

A feature definition does not necessarily provide all the information required to properly implement glyph substitution or positioning actions. Often, a text-processing client may need to supply additional data<sup>1</sup> In all cases, the text-processing client is responsible for applying, combining, and arbitrating among features and rendering the result.

The list of features registered by Microsoft together with a short description follows.<sup>2</sup>

<b>aalt</b>	Access All Alternates	<b>falt</b>	Final Glyph on Line Alternates	<b>liga</b>	Standard Ligatures
<b>abvf</b>	Above-base Forms			<b>ljmo</b>	Leading Jamo Forms
<b>abvm</b>	Above-base Mark Positioning	<b>fin2</b>	Terminal Forms #2	<b>lnum</b>	Lining Figures
		<b>fin3</b>	Terminal Forms #3	<b>loc1</b>	Localized Forms
<b>abvs</b>	Above-base Substitutions	<b>fin4</b>	Terminal Forms	<b>mark</b>	Mark Positioning
<b>afrs</b>	Alternative Fractions	<b>frac</b>	Fractions	<b>med2</b>	Medial Forms #2
<b>akhn</b>	Akhands	<b>fwid</b>	Full Widths	<b>medi</b>	Medial Forms
<b>blwf</b>	Below-base Forms	<b>half</b>	Half Forms	<b>mgrk</b>	Mathematical Greek
<b>blwm</b>	Below-base Mark Positioning	<b>haln</b>	Halant Forms	<b>mkmk</b>	Mark to Mark Positioning
		<b>halt</b>	Alternate Half Widths	<b>mset</b>	Mark Positioning via Substitution
<b>blws</b>	Below-base Substitutions	<b>hist</b>	Historical Forms		
<b>calt</b>	Contextual Alternates	<b>hkna</b>	Horizontal Kana Alternates	<b>nalt</b>	Alternate Annotation Forms
<b>case</b>	Case-Sensitive Forms	<b>hlig</b>	Historical Ligatures	<b>nlck</b>	NLC Kanji Forms
<b>ccmp</b>	Glyph Composition and Decomposition	<b>hngl</b>	Hangul	<b>nukt</b>	Nukta Forms
<b>clig</b>	Contextual Ligatures	<b>hojo</b>	Hojo Kanji Forms (JIS X 0212-1990 Kanji Forms)	<b>numr</b>	Numerators
<b>cpSP</b>	Capital Spacing	<b>hwid</b>	Half Widths	<b>onum</b>	Oldstyle Figures
<b>cswh</b>	Contextual Swash	<b>init</b>	Initial Forms	<b>opbd</b>	Optical Bounds
<b>curs</b>	Cursive Positioning	<b>isol</b>	Isolated Forms	<b>ordn</b>	Ordinals
<b>c2sc</b>	Small Capitals From Capitals	<b>ital</b>	Italics	<b>ornm</b>	Ornaments
<b>c2pc</b>	Petite Capitals From Capitals	<b>jalt</b>	Justification Alternates	<b>palt</b>	Proportional Alternate Widths
<b>dist</b>	Distances	<b>jp78</b>	JIS78 Forms	<b>pcap</b>	Petite Capitals
<b>dlig</b>	Discretionary Ligatures	<b>jp83</b>	JIS83 Forms	<b>pnum</b>	Proportional Figures
<b>dnom</b>	Denominators	<b>jp90</b>	JIS90 Forms	<b>pref</b>	Pre-Base Forms
		<b>jp04</b>	JIS2004 Forms	<b>pres</b>	Pre-base Substitutions
<b>expt</b>	Expert Forms	<b>kern</b>	Kerning	<b>pstf</b>	Post-base Forms
		<b>lfbf</b>	Left Bounds		

<sup>1</sup>As an example let us consider the `init` feature whose function is to provide initial glyph forms. Nothing in the feature's lookup tables indicates when or where to apply this feature during text processing. Hence, to correctly use this feature in Arabic text where initial glyph forms appear at the beginning of words, text-processing clients must be able to identify the first glyph position in each word before making the glyph substitution.

<sup>2</sup>More details about each feature are available at the Microsoft OpenType site <http://www.microsoft.com/typography/otspec/featuretags.htm>, or Adobe developers' site [http://partners.adobe.com/public/developer/opentype/index\\_tag3.html](http://partners.adobe.com/public/developer/opentype/index_tag3.html)

<b>psts</b>	Post-base Substitutions	<b>ss06</b>	Stylistic Set 6	<b>tjmo</b>	Trailing Jamo Forms
<b>pwid</b>	Proportional Widths	<b>ss07</b>	Stylistic Set 7	<b>tnam</b>	Traditional Name Forms
<b>qwid</b>	Quarter Widths	<b>ss08</b>	Stylistic Set 8	<b>tnum</b>	Tabular Figures
<b>rand</b>	Randomize	<b>ss09</b>	Stylistic Set 9	<b>trad</b>	Traditional Forms
<b>rlig</b>	Required Ligatures	<b>ss10</b>	Stylistic Set 10	<b>twid</b>	Third Widths
<b>rphf</b>	Reph Forms	<b>ss11</b>	Stylistic Set 11	<b>unic</b>	Unicase
<b>rtbd</b>	Right Bounds	<b>ss12</b>	Stylistic Set 12	<b>valt</b>	Alternate Vertical Metrics
<b>rtla</b>	Right-to-left alternates	<b>ss13</b>	Stylistic Set 13	<b>vatu</b>	Vattu Variants
<b>ruby</b>	Ruby Notation Forms	<b>ss14</b>	Stylistic Set 14	<b>vert</b>	Vertical Writing
<b>salt</b>	Stylistic Alternates	<b>ss15</b>	Stylistic Set 15	<b>vhal</b>	Alternate Vertical Half Metrics
<b>sinf</b>	Scientific Inferiors	<b>ss16</b>	Stylistic Set 16	<b>vjmo</b>	Vowel Jamo Forms
<b>size</b>	Optical size	<b>ss17</b>	Stylistic Set 17	<b>vkna</b>	Vertical Kana Alternates
<b>smcp</b>	Small Capitals	<b>ss18</b>	Stylistic Set 18	<b>vkra</b>	Vertical Kerning
<b>smp1</b>	Simplified Forms	<b>ss19</b>	Stylistic Set 19	<b>vpa1</b>	Proportional Alternate Vertical Metrics
<b>ss01</b>	Stylistic Set 1	<b>ss20</b>	Stylistic Set 20	<b>vrt2</b>	Vertical Alternates and Rotation
<b>ss02</b>	Stylistic Set 2	<b>subs</b>	Subscript	<b>zero</b>	Slashed Zero
<b>ss03</b>	Stylistic Set 3	<b>sup</b>	Superscript		
<b>ss04</b>	Stylistic Set 4	<b>swsh</b>	Swash		
<b>ss05</b>	Stylistic Set 5	<b>titl</b>	Titling		

### 21.1.4.3 OpenType support today

As an example of how publishing applications can exploit OpenType's layout features we can look at OpenType support in Adobe's Illustrator, InDesign and Photoshop<sup>1</sup> programs. These include automatic substitution by alternate glyphs in an OpenType Pro font (ligatures, small capitals, and proportional old-style figures, vertical shift of punctuation in an all-caps setting). Moreover, any alternate glyphs in OpenType fonts may be selected manually via the *Insert Character* palette (see Figure 21.1). These OpenType Pro fonts offer a full range of accented characters to support all central and eastern European languages, and many of them also contain support for the Cyrillic and Greek alphabets.

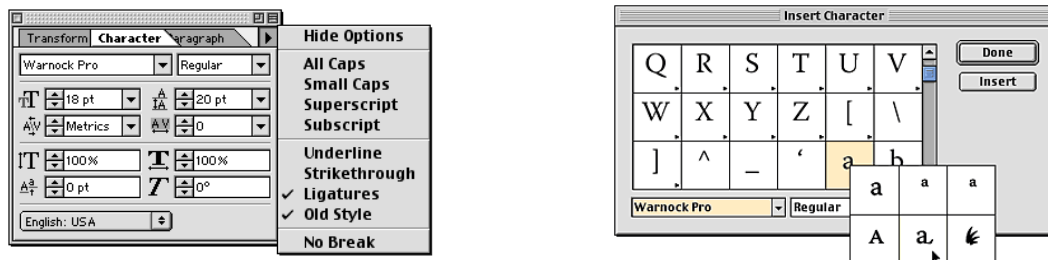


Figure 21.1: Using OpenType's advanced typographic features in Adobe InDesign. Left: selection of automatic substitution of ligatures and old-style figures on a menu. Right: select and insert any alternate glyph *Insert Character* palette.

Feature support across Microsoft's Office applications exists for those features that are necessary for language support, such as contextual substitutions for Arabic—and only in the languages which require them (e.g., Word 2003 does contextual substitutions for Arabic, but not for English).

Openoffice on all supported platforms has a somewhat similar approach to Microsoft's Office suite

<sup>1</sup>See <http://www.adobe.com/products/XXX/main.htm>, where XXX stands for *illustrator*, *indesign*, and *photoshop*, respectively.

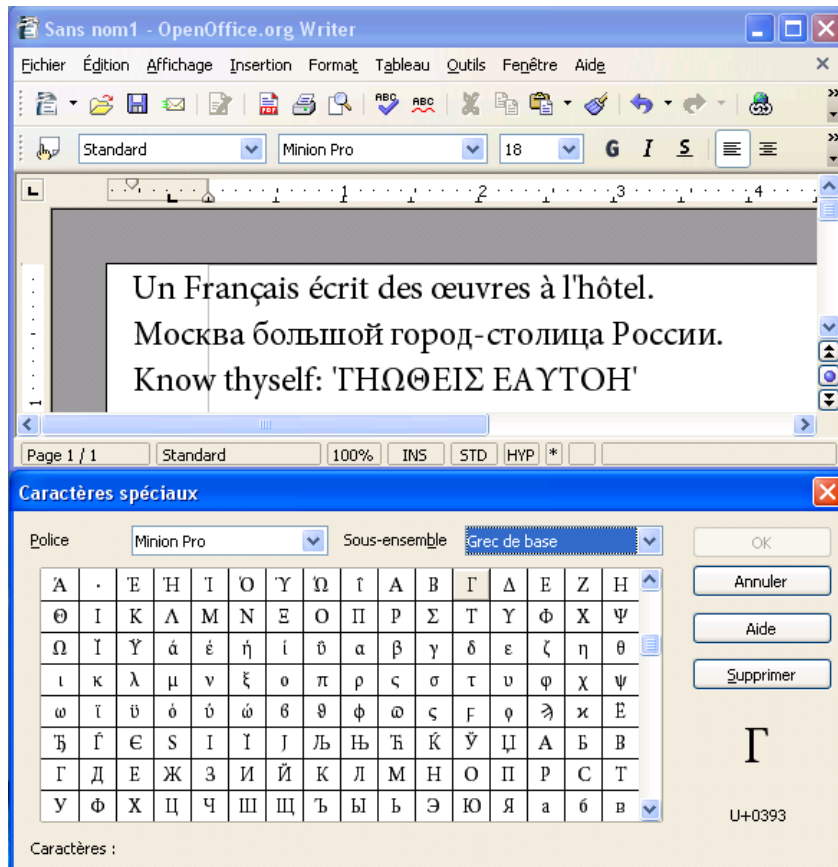


Figure 21.2: OpenType Unicode support in OpenOffice. The top panel shows text in various alphabets and the bottom panel the characters available in the Greek part of font layout.

in that it allows one to use the characters present in the font but does not really present an interface to the advanced typographic features (see Figure 21.2).

That leaves us with the availability of the fonts themselves. Around the year 2000 there were only a handful of OpenType fonts, and almost all of them were from Adobe. Nowadays, there are thousands available from over two dozen font foundries. For instance, the entire Adobe Type Library of over 2,200 fonts has been translated into the OpenType format, URW has released over 1,000 OpenType fonts, and other large foundries, such as Linotype and Agfa Monotype, as well as most smaller foundries, are also creating OpenType fonts. Most of Microsoft's system fonts, and Apple's Japanese system fonts, are OpenType. Similarly, OpenType is being embraced by major type foundries for non-alphabetic scripts, such as Chinese and Japanese.

However, it is not enough for a font to be in the OpenType format to be sure that it has extended language support or extra typographic features. Therefore, before purchasing, you should examine the features present in a font.<sup>1</sup> To inspect a font that you already have on your Microsoft Windows system, you can install the *Font Properties Extension* from Microsoft. This add-on allows you to right-click on a font to display a much expanded set of properties, which includes language support and OpenType

<sup>1</sup>In the case of Adobe, where currently not all fonts released in OpenType format have significant added features or extended language support, you browse all fonts in the *Adobe Type Library* from the URL <http://store.adobe.com/type/main.html>, so that you can inspect the font you are interested in. Other font vendors offer similar possibilities.



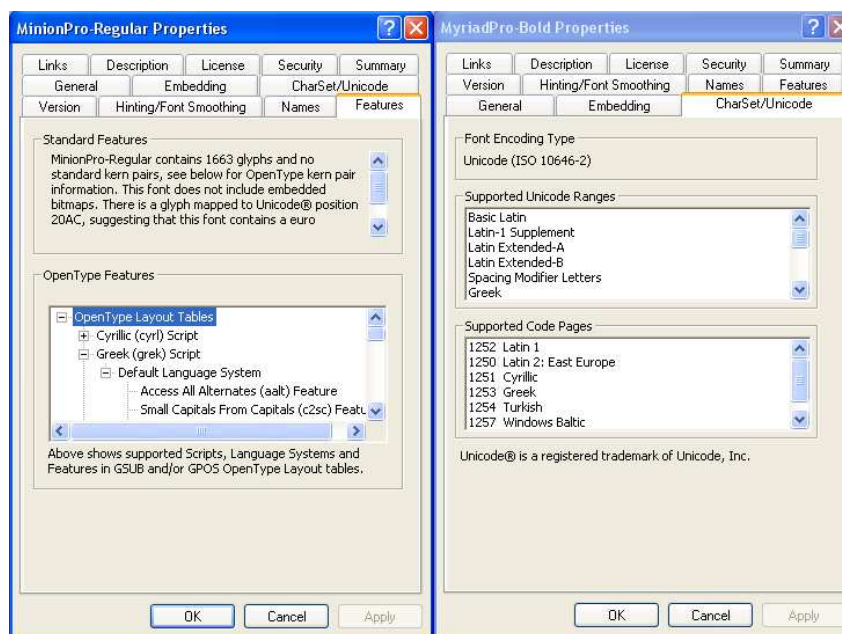


Figure 21.3: Microsoft's *Fonts Extension* utility displays OpenType features for MinionPro-Regular and the supported Character sets for MyriadPro-Bold when you right-click on the font (This utility, `ttfont`, adds several new property tabs to the standards properties dialog box, such as information relating to font origination and copyright, the type sizes to which hinting and smoothing are applied, and the code pages supported by extended character. It can be downloaded from <http://www.microsoft.com/typography/TrueTypeProperty21.msp>.)

layout features (see Figure 21.3).

#### 21.1.4.4 Interrogating OpenType fonts

Eddie Kohler's `otfinfo` program<sup>1</sup> prints information about an OpenType font.

```
> otfinfo --help
'Otfinfo' reports information about an OpenType font to standard output.
Options specify what information to print.

Usage: otfinfo [-sfzpg] [OTFFILES...]

Query options:
  -s, --scripts           Report font's supported scripts.
  -f, --features          Report font's GSUB/GPOS features.
  -z, --optical-size      Report font's optical size information.
  -p, --postscript-name   Report font's PostScript name.
  -a, --family            Report font's family name.
  -v, --font-version      Report font's version information.
  -i, --info              Report font's names and designer/vendor info.
  -g, --glyphs            Report font's glyph names.
  -t, --tables            Report font's OpenType tables.
```

<sup>1</sup>Part of his `lcdf` tools, see [www.lcdf.org/type/](http://www.lcdf.org/type/).



```

Other options:
  --script=SCRIPT[.LANG]  Set script used for --features [latn].
  -V, --verbose           Print progress information to standard error.
  -h, --help             Print this message and exit.
  -q, --quiet            Do not generate any error messages.
  --version              Print version number and exit.

> otfinfo --info texmf-commercial/fonts/opentype/adobe/minionpro-regular.otf
Family:                Minion Pro
Subfamily:             Regular
Full name:             Minion Pro
PostScript name:      MinionPro-Regular
Version:              Version 2.012;PS 002.000;Core 1.0.38;makeotf.lib1.6.6565
Unique ID:            2.012;ADBE;MinionPro-Regular
Designer:             Robert Slimbach
Vendor URL:           http://www.adobe.com/type/
Trademark:            Minion is either a ...
Copyright:            © 2000, 2002, 2004 ...
License URL:          http://www.adobe.com/type/legal.html

> otfinfo --script texmf-commercial/fonts/opentype/adobe/minionpro-regular.otf
cyr1      Cyrillic      latn.DEU   Latin/German (Standard)
grek      Greek         latn.MOL   Latin/Moldavian
latn      Latin         latn.ROM   Latin/Romanian
latn.AZE  Latin/Azeri     latn.SRB   Latin/Serbian
latn.CRT  Latin/Crimean Tatar latn.TRK   Latin/Turkish

> otfinfo --tables texmf-commercial/fonts/opentype/adobe/minionpro-regular.otf
    64 BASE                54 head
132417 CFF                 36 hhea
    5228 DSIG              6652 hmtx
    40074 GPOS              6 maxp
    13872 GSUB             1533 name
    96 OS/2                32 post
    4048 cmap

> otfinfo --features texmf-commercial/fonts/opentype/adobe/minionpro-regular.otf
aalt      Access All Alternates      c2sc      Small Capitals From Capitals
case      Case-Sensitive Forms        cpsp      Capital Spacing
dlig      Discretionary Ligatures     dnom      Denominators
fina      Terminal Forms              frac      Fractions
hist      Historical Forms            kern      Kerning
liga      Standard Ligatures          lnum      Lining Figures
numr      Numerators                 onum      Oldstyle Figures
ordn      Ordinals                   ornm      Ornaments
pnum      Proportional Figures        salt      Stylistic Alternates
sinf      Scientific Inferiors        size      Optical Size
smcp      Small Capitals              ss01      Stylistic Set 1
ss02      Stylistic Set 2            sups      Superscript
tnum      Tabular Figures             zero      Slashed Zero

```

Just van Rossum's `ttx` utility<sup>1</sup> can decompile the contents of an OpenType font and output it in XML format. This comes in handy if you want to study the contents of a given font (e.g., its tables) or (slightly) modify it.

```
> ttx --help
usage: ttx [options] inputfile1 [... inputfileN]

TTX 2.0b1 -- From OpenType To XML And Back

If an input file is a TrueType or OpenType font file, it will be
dumped to an TTX file (an XML-based text format).
If an input file is a TTX file, it will be compiled to a TrueType
or OpenType font file.

Output files are created so they are unique: an existing file is
never overwritten.

General options:
-h Help: print this message
-d <outputfolder> Specify a directory where the output files are
  to be created.
-v Verbose: more messages will be written to stdout about what
  is being done.

Dump options:
-l List table info: instead of dumping to a TTX file, list some
  minimal info about each table.
-t <table> Specify a table to dump. Multiple -t options
  are allowed. When no -t option is specified, all tables
  will be dumped.
-x <table> Specify a table to exclude from the dump. Multiple
  -x options are allowed. -t and -x are mutually exclusive.
-s Split tables: save the TTX data into separate TTX files per
  table and write one small TTX file that contains references
  to the individual table dumps. This file can be used as
  input to ttx, as long as the table files are in the
  same directory.
-i Do NOT disassemble TT instructions: when this option is given,
  all TrueType programs (glyph programs, the font program and the
  pre-program) will be written to the TTX file as hex data
  instead of assembly. This saves some time and makes the TTX
  file smaller.

Compile options:
-m Merge with TrueType-input-file: specify a TrueType or OpenType
  font file to be merged with the TTX file. This option is only
  valid when at most one TTX file is specified.
-b Don't recalc glyph bounding boxes: use the values in the TTX
  file as-is.
```

Thus, to decompile a font `myfont.otf` just specify:

```
> ttx myfont.otf
```

---

<sup>1</sup>Written in Python and part of the FontTools toolset ([sourceforge.net/projects/fonttools](http://sourceforge.net/projects/fonttools)).

This will write a file `myfont.ttf` in the directory where the font file resides. If you are only interested in two tables (e.g., GSUB and GPOS), specify them on the command line:

```
> ttx -t GSUB -t GPOS myfont.otf
```

To convert an XML file `myfont.ttf` back into an OpenType or TrueType file is similarly easy:

```
> ttx myfont.ttf
```

If you want to introduce modifications (e.g., given in XML format in the file `myfontmods.ttf`) into an OpenType file, use the `-m` option, as follows:

```
> ttx -m myfont.otf myfontmods.ttf
```

An more explicit example with the font MinionPro follows.

```
> ttx -l /texlive/2007/texmf-commercial/fonts/opentype/adobe/minionpro-regular.otf
```

Listing table info for "/texlive/2007/texmf-commercial/fonts/opentype/adobe/minionpro-regular.otf":

tag	checksum	length	offset	tag	checksum	length	offset
----	-----	-----	-----	----	-----	-----	-----
BASE	0x086729a7	64	199052	CFF	0x101232c2	132417	6032
DSIG	0x446dbd94	5228	199116	GPOS	0xx71552700	40074	158976
GSUB	0xx3bf7bcba	13872	145104	OS/2	0x40e57e9f	96	320
cmap	0x0cedc8f1	4048	1952	head	0xx2167aded	54	220
hhea	0x09140bb5	36	276	hmtx	0xx37425493	6652	138452
maxp	0x067f5000	6	312	name	0x3cf7b183	1533	416
post	0x0x47ffce	32	6000				

```
ttx -d. -t head /texlive/2007/texmf-commercial/fonts/opentype/adobe/minionpro-regular.otf
```

Dumping "/texlive/2007/texmf-commercial/fonts/opentype/adobe/minionpro-regular.otf" to "./minionpro-

Dumping 'head' table...

```
> less ./minionpro-regular.ttf
```

```
<?xml version="1.0" encoding="ISO-8859-1"?>
```

```
<ttFont sfntVersion="OTTO" ttLibVersion="2.0b1">
```

```
<head>
```

```
<!-- Most of this table will be recalculated by the compiler -->
```

```
<tableVersion value="1.0"/>
```

```
<fontRevision value="2.0119934082"/>
```

```
<checksumAdjustment value="-0x107d913c"/>
```

```
<magicNumber value="0x5f0f3cf5"/>
```

```
<flags value="00000000 00000011"/>
```

```
<unitsPerEm value="1000"/>
```

```
<created value="Tue Jun 29 11:41:10 2004"/>
```

```
<modified value="Tue Jun 29 11:41:10 2004"/>
```

```
<xMin value="-290"/>
```

```
<yMin value="-360"/>
```

```
<xMax value="1684"/>
```

```
<yMax value="989"/>
```

```
<macStyle value="00000000 00000000"/>
```

```
<lowestRecPPEM value="3"/>
```

```
<fontDirectionHint value="2"/>
```

```
<indexToLocFormat value="0"/>
```

```
<glyphDataFormat value="0"/>
```

```
</head>
```

```
</ttFont>
```

For reasons of efficiency TrueType and OpenType font instances can be grouped into “collection” (`.ttc`), so that different fonts can share common tables to describe glyphs. Some programs are not able to extract the various font components from such a collection. To help with this problem a small utility, `ttc2ttf`, exists to extract the font instances from a collection.

## 21.2 Typography: combining characters for optimal readability

A typesetting system has the task to combine the characters that are input to the system into lines and paragraphs. The typeset result should look pleasing and readable. Various approaches have been used to address this task. Most of them are heuristical, whereas some are more mathematical, in particular  $\TeX$ . It is not our purpose to describe in any detail the complex typesetting algorithms of  $\TeX$ [11].<sup>1</sup> We merely want to give some basic information on how characters are put together on a line.

### 21.2.1 The font metric files

To introduce the main aspects of font metrics we use the Adobe Font Metric (`.afm`) file for PostScript fonts as an example.  $\TeX$ 's font metric files (`.tfm`) and the tables of TrueType and OpenType fonts contain similar information.

#### 21.2.1.1 The structure of an AFM file

AFM files,<sup>2</sup> used as a standard means of interchanging font metric information between programs, are machine-independent and extensible ASCII-encoded files that are characterized by the `.afm` file extension. An `.afm` file provides both global metrics for a font program and information for individual characters.

Let us have a closer look at the `.afm` file of the font *Times-Bold*. In general, an `.afm` file has four parts. The *first* part has general information about the font, with its name, weight, cap height, x-height, and default encoding:

```
StartFontMetrics 2.0
Comment Creation Date: Fri Sep 6 17:58:27 1991
Comment UniqueID 28417
Comment VMusage 30458 37350
FontName Times-Bold
FullName Times Bold
FamilyName Times
Weight Bold
ItalicAngle 0
IsFixedPitch false
FontBBox -168 -218 1000 935
UnderlinePosition -100
UnderlineThickness 50
Version 001.007
Notice Copyright (c) 1991 Adobe Systems Incorporated.
EncodingScheme AdobeStandardEncoding
CapHeight 676
```

<sup>1</sup>The web page <http://www.leverkruid.nl/GKPLinebreaking/elements-xhtml.xml> describes recent research by Simon Pepping to generalize  $\TeX$ 's Knuth–Plass linebreaking algorithm.

<sup>2</sup>See [http://partners.adobe.com/public/developer/en/font/5004.AFM\\_Spec.pdf](http://partners.adobe.com/public/developer/en/font/5004.AFM_Spec.pdf), the *Adobe Font Metric Format Specification*.

```
XHeight 461
Ascender 676
Descender -205
```

The dimensions of the characters are given on a notional grid of 1000 units square to which all fields relate. In particular, the `FontBBox` entry in the header information tells us that all characters of the font stay within a range of  $[-168, 1000]$  units in  $x$  and  $[-218, 935]$  units in  $y$ , the x-height (`XHeight`, the height of lowercase letters, e.g., “x”) is 461 units, the height of capitals (`CapHeight`, e.g., “M”) and large lowercase letters (`Ascender`, e.g., “h”) is 676 units, etc.

The global header information is followed by the *second* part, a set of metric fields, one for each character in the font. A field contains:

- C** a numeric code; this is the position in the default Adobe encoding (characters with a code of  $-1$ , such as `egrave` in the following example, are not available without re-encoding);
- W** the width of the character;
- N** the name of the character;
- B** the character bounding box;
- L** possible ligature combinations (optional).

A few lines of the character metrics part of the `.afm` file follow.

```
1 StartCharMetrics 228
2 C 32 ; WX 250 ; N space ; B 0 0 0 0 ;
3 C 39 ; WX 333 ; N quoteright ; B 79 356 263 691 ;
4 C 44 ; WX 250 ; N comma ; B 39 -180 223 155 ;
5 C 46 ; WX 250 ; N period ; B 41 -13 210 156 ;
6 C 65 ; WX 722 ; N A ; B 9 0 689 690 ;
7 C 70 ; WX 611 ; N F ; B 16 0 583 676 ;
8 C 77 ; WX 944 ; N M ; B 14 0 921 676 ;
9 C 86 ; WX 722 ; N V ; B 16 -18 701 676 ;
10 C 102 ; WX 333 ; N f ; B 14 0 389 691 ; L i fi ; L l fl ;
11 C 103 ; WX 500 ; N g ; B 28 -206 483 473 ;
12 C 105 ; WX 278 ; N i ; B 16 0 255 691 ;
13 C 108 ; WX 278 ; N l ; B 16 0 255 676 ;
14 ...
15 C -1 ; WX 444 ; N egrave ; B 25 -14 426 713 ;
16 EndCharMetrics
```

Let us now see how we can use this information for combining individual characters into “words”. Figure 21.4 on the next page shows a few characters of the PostScript fonts *Times-Bold* (top) and *Times-BoldItalic* (bottom) drawn at 350 points. For reference, the grid shows units in PostScript points. The *baseline* is the line upon which most letters “sit” and under which descenders extend. In our case the baseline of the top line is at a  $y$  value of 500 and that of the bottom line at 100.

Each character is surrounded by its “bounding box,” i.e., the largest imaginary box that can be drawn around the glyph which just touches on all sides (the coordinates of the lower left and upper right corners of each character’s bounding box starts with the `B` key on the lines above). Take the first character, the “period,” which we placed at a value of (50, 500) on the grid. We have marked its origin with a cross. The next character will then be placed, ignoring kerning, see below, one “width” (`WX`) to the right. For the “period” character (line 5) this is 250 units. This value should be multiplied by  $350/1000$  (the ratio of the font size to the internal unit, which is 1000), i.e., yielding 87.5, so that the “M” character starts at 137.5 (second cross on top line of figure). We proceed like this to place each character on the line. Characters do not precisely start at their reference point, but somewhat to the right. For instance, the “g” on the top line has a *leftsidebearing* (the first value following the `B` key on

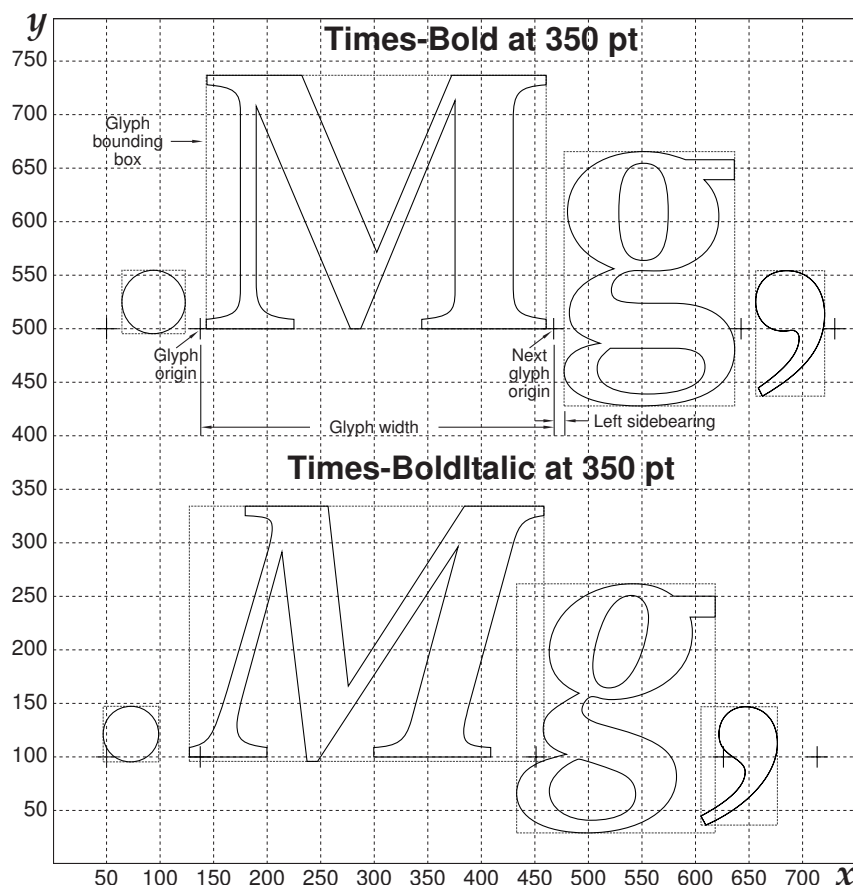


Figure 21.4: Font bounding boxes at work

line 11) of 28 units (9.8 on the grid, as indicated in the figure). Similarly the “period” (line 5), which starts the line, is offset by 48 units (16.8 on the grid) from its reference point.

Character bounding boxes can overlap, which is more easily seen in the case of an italic font, e.g., *Times-BoldItalic* in the bottom line of Figure 21.4. We reproduce here the `.afm` entries corresponding to the four characters in that figure.

```
C 44 ; WX 250 ; N comma ; B -60 -182 144 134 ;
C 46 ; WX 250 ; N period ; B -9 -13 139 135 ;
C 77 ; WX 889 ; N M ; B -29 -12 917 669 ;
C 103 ; WX 500 ; N g ; B -52 -203 478 462 ;
```

The left bearings (the first value following the `B` key above) of all characters are negative and the widths of the letters is smaller than the horizontal dimension of their bounding box, so that the bounding box of the next character starts to the left of the rightmost part of the current letter. This is especially true for the “Mg” combination, where the descender of the “g” lies firmly to the left of the right top of the “M.”

The effect of bounding box positioning and how it depends on the series and shape of the font family is also clearly observed in Figure 21.5 on the next page, where for variants of *Times Roman* (from top to bottom we have *Times-Roman*, *Times-Italic*, *Times-Bold*, and *Times-BoldItalic*) we show a

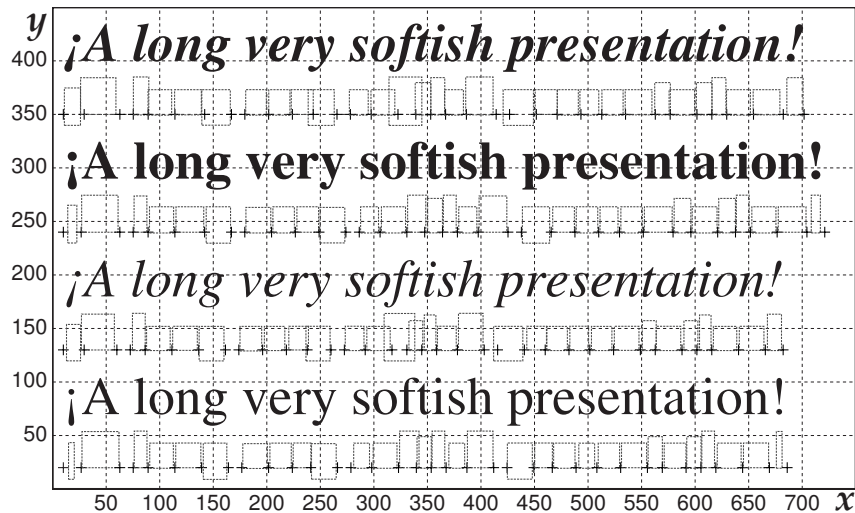


Figure 21.5: Putting bounding boxes on a line

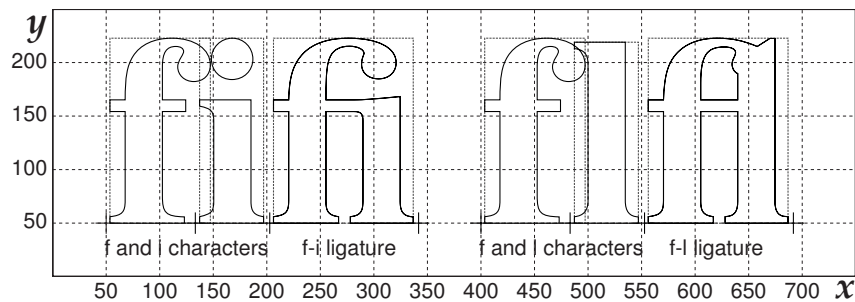


Figure 21.6: Two common ligatures

short text line, together with the bounding boxes and reference points of the individual characters. Note in particular how the bounding boxes in the “*fi*” combination for the italic variant overlap.

This brings us naturally to the notion of *ligature*. In high-quality typography, a ligature occurs where two or more letterforms are written or printed as a unit. Generally, ligatures replace two or more characters that occur next to each other with a single glyph. Often such characters share common components.<sup>1</sup>

One of the most common ligatures is “fi,” where the dot above a lowercase “i” interferes with the loop on the lowercase “f”. Therefore, the two-letter combination “fi” is replaced by a single entity with the dot absorbed into the “f” (see left part of Figure 21.6.) Here are three lines from the .afm file that have to do with the “f” character.

```
C 102 ; WX 333 ; N f ; B 14 0 389 691 ; L i fi ; L l fl ;
C 174 ; WX 556 ; N fi ; B 14 0 536 691 ;
C 175 ; WX 556 ; N fl ; B 14 0 536 691 ;
```

<sup>1</sup>Ligatures are a subset of so-called *contextual forms*, where the particular rendering of a letter combination depends upon its context, such as surrounding letters or whether or not it appears at the end of a line, as in Arabic.

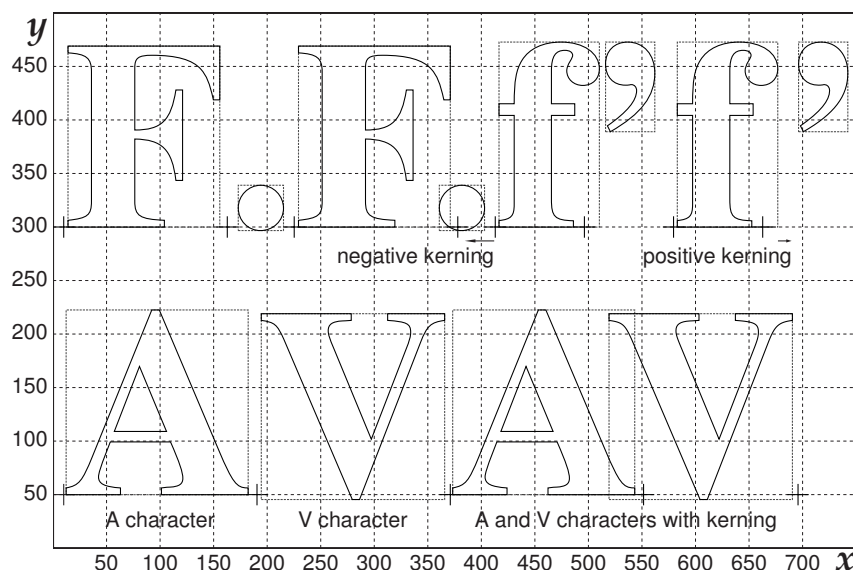


Figure 21.7: The mechanics of kerning

The first line states (`L` keyword) that a “f” followed by a “i” should be replaced by the character with name “fi” (line 2) while a “f” followed by a “l” should be replaced by the character with name “fl” (line 3). The latter ligature is seen at the right of Figure 21.6 on the previous page.

The font family *Times Roman* contains only the “fi” and “fl” ligatures, but several fonts offer many more (e.g., the font *Minion Pro* as displayed in Table 21.6 contains ligatures for “fb”, “ff”, “ffb”, “ffh”, “ffi”, “ffj”, etc.).

The *third* part of an `.afm` file consists of *kerning* information. Kerning is the process of adjusting letter spacing in a proportional font, so that the level of grey along a line of text is as homogeneously distributed as possible. Kerning values can be positive (increase space between character pairs) or negative (decrease space).

In PostScript kerning is limited to pairs of individual characters, whereas with OpenType the kerning algorithm is generalized to the use of kerning classes, where one offset is stored for any pair of characters from two sets, for example (V,W) and (a,e,o). The use of kerning classes is made necessary by the fact that nowadays fonts come often with several thousands of glyphs for many languages so that the number of kerning pairs to be specified, especially for accented letters, would become unmanageable.

Let us extract the kerning information for the characters in Figure 21.7 from the `.afm` file of *Times-Bold*.

```
StartKernData
StartKernPairs 496
...
KPX A V          -145
KPX F period     -110
KPX f quoteright 55
...
EndKernPairs
EndKernData
```

The letters in Figure 21.7 are drawn at 250 pt, so that the kerning figures above should be multiplied



by 250/1000 to translate them to grid coordinates. Let us start with the bottom line, typical of capital letters with oblique stems. In this case we are told to move the “A” and “V” characters together by an amount of  $-145$  units (i.e., on our grid the “V” is kerned 36.25 pt to the left.) Often punctuation following capitals has also to be kerned, as shown in the first part of the top line in the figure, where the “period” is kerned 110 units negatively (the period moves 27.5 pt left below the arm of the “F”, as shown by the arrow). On the other hand, quotes interfere with the upper final parts of letters with ascenders, as here with the “f”. Thus we move the quote out towards the right by a positive kern of 55 units (i.e., 13.75 pt as shown by the little arrow pointing right in our figure).

The *fourth* and last part of the `.afm` file contains information about composite characters and shows how they are to be constructed; for example, `Aacute` is not a character in its own right, but is formed by joining an `A` and an acute accent.

```
StartComposites 58
CC Aacute      2 ; PCC A 0 0 ; PCC acute      188 210 ;
CC Acircumflex 2 ; PCC A 0 0 ; PCC circumflex 188 210 ;
CC Adieresis   2 ; PCC A 0 0 ; PCC dieresis   188 210 ;
CC Agrave      2 ; PCC A 0 0 ; PCC grave      188 210 ;
....
EndComposites
EndFontMetrics
```

The third and fourth parts of an `.afm` file are optional; a math symbol font would simply have the general information and the individual character metrics.

### 21.2.1.2 The $\TeX$ font metrics (`.tfm`)

The metric information that  $\TeX$  needs about characters of a font is stored in files with the extension `.tfm` (for  $\TeX$  font metric). For each character in a font four dimensions are recorded: its width, its height above and its depth below the baseline, and its “italic correction”, i.e., the amount of extra space that is added with the `\/` command to take into account a switch from a slanted to an upright typeface.<sup>1</sup> This information is similar to the width and bounding box information present in an `.afm` file.

The `.tfm` format also contains data about relationships between characters, such as kernings and ligatures. Finally, again as in an `.afm` file, the `.tfm` format includes information about the font as a whole: a number of dimensions denoting the font design size, the width of a normal space, its stretchability, and so on. From within  $\TeX$  they are accessible via the `\fontdimen` command [15, p. 428]. Fonts for mathematics have extra dimensions for specialized typesetting.

For reasons of efficiency the information in the `.tfm` file is stored in binary format. The file can be made human-readable with the help of the program `tftopl`, which produces a `.pl` (*property list*) file that can be modified, if necessary. Conversely, the `pltotf` program transforms a `.pl` file into a `.tfm` file.

## 21.2.2 $\TeX$ virtual fonts

Virtual fonts [10] are like font metric files in that they contain general information about the font and detailed instructions for each character. But they stand at a higher level: they describe an “imaginary” font that can consist of bits of other fonts, low-level `.dvi` code such as line-drawing, or even `\special` commands. Thus character 47 may say “set a capital T from Times Roman”, character 53 may say “set a 20 point = sign from Computer Modern”, and character 101 may say “use a `\special` command to include this logo picture”. Virtual fonts are used to pretend to  $\TeX$  that another font is arranged like

<sup>1</sup> $\TeX$  takes automatically care of this by adding `\/` commands in the right places in its high-level commands, such as `\textit` and `\emph`.

a Computer Modern font, to allow all T<sub>E</sub>X's internal macros to work without problem, or to combine specialized characters from an “expert” font with a normal text font.

A virtual font has two parts: a `.tfm` metric file that T<sub>E</sub>X reads, and a `.vf` file that drivers read and that tells them how to create each character. Most often, `.vf` files are used simply to rearrange a font (i.e., “to create character 212 in this font, take character 176 from the other font”), but it can also be used to combine two fonts, join arbitrary characters, or even change color, rotate characters or perform other operations.

Just like `.tfm` files, virtual fonts have a human-readable form—the `.vpl` file. Conversion between `.vf` and `.vpl` format is handled by the programs `vftovp` and `vptovf`.

In addition to property entries that are allowed in `.pl` files, a `.vpl` file can contain property types that deal with selecting characters from one or more fonts (for which `.tfm` or `.vf` files must exist; selecting characters recursively is allowed). For instance, `VTITLE` identifies the virtual font (default value is an empty string), the `MAPFONT` property, declares the fonts from which characters can be selected, and the `MAP` property, which can appear as part of a `CHARACTER` property, specifies where the character should be taken from. The last two properties take as values property lists with properties relevant only for `.vf` files. Let us look at the virtual font for *Times Roman* constructed to simulate small capitals (by scaling down the base font by 80%). The base font `ptmr8r.tfm` (T<sub>E</sub>XBase1 encoding) has been reencoded to the T1 encoding (`ptmr8t.tfm`) and then scaled down to small capitals `ptmrc8t.tfm` and `ptmrc8t.vf` (see Section 21.7 for an explanation of this font naming convention). Let us create a human-readable `.vpl` file:

```
> vftovp ptmrc8t.vf ptmrc8t.tfm ptmrc8t.vpl
```

The output file `ptmrc8t.vpl` starts as follows.

```
(VTITLE )
(FAMILY UNSPECIFIED)
(FACE F MRR)
(CODINGScheme EXTENDED TEX FONT ENCODING - LATIN)
(DESIGNSIZE R 10.0)
(COMMENT DESIGNSIZE IS IN POINTS)
(COMMENT OTHER SIZES ARE MULTIPLES OF DESIGNSIZE)
(CHECKSUM O 27461762233)
(FONTDIMEN
  (SLANT R 0.0)
  (SPACE R 0.25)
  \emph{... several lines deleted}
)
(MAPFONT D 0
  (FONTNAME ptmr8r)
  (FONTCHECKSUM O 4767720433)
  (FONTAT R 0.8)
  (FONTSIZE R 10.0)
)
(MAPFONT D 1
  (FONTNAME ptmr8r)
  (FONTCHECKSUM O 4767720433)
  (FONTAT R 1.0)
  (FONTSIZE R 10.0)
)
```

Inside a `MAPFONT` property, the `FONTNAME` refers to the relevant `.tfm` file; `FONTCHECKSUM` should match the checksum in that font so that the processing software can tell if the right file was found.

`FONTDSIZE` should match the design size of the font (10 pt in this case) and `FONTAT` specifies a scaling factor (0.8 for font “0”, which implements artificial small caps).

A simple entry (here for the lowercase letter “e”) gives the width and height then picks the uppercase “e” from the scaled (default) font to construct the small capital.

```
(CHARACTER C e
  (CHARWD R 0.537988)
  (CHARHT R 0.514496)
  (MAP
    (MOVERIGHT R 0.025)
    (SETCHAR C E)
    (MOVERIGHT R 0.025)
  )
)
```

A more interesting example is the German “ß”, which should be turned into two small capital “S” characters. In the T1 encoding the “ß” character has octal value 377 (see Table 21.15 in Appendix 21.8) and that is replaced by a succession of two (scaled down) “S” characters from font “0”. On the other hand the uppercase “SS” (octal code 337, the second entry below) is an unscaled (`SELECTFONT 1` for font “1”) combination of two uppercase “S” characters, as expected.

```
(CHARACTER O 377
  (CHARWD R 0.937988)
  (CHARHT R 0.536994)
  (CHARDP R 0.008991)
  (MAP
    (MOVERIGHT R 0.025)
    (SETCHAR C S)
    (SETCHAR C S)
    (MOVERIGHT R 0.025)
  )
)
(CHARACTER O 337
  (CHARWD R 1.212)
  (CHARHT R 0.672992)
  (CHARDP R 0.008991)
  (MAP
    (MOVERIGHT R 0.025)
    (SELECTFONT D 1)
    (SETCHAR C S)
    (MOVERIGHT R 0.05)
    (SETCHAR C S)
    (MOVERIGHT R 0.025)
  )
)
```

In principle *anything* that is valid in a `dvi` file can be used in a virtual font, for instance `\special` commands, that can include raw PostScript code. Note, however, that such code is in general highly nonportable, since it depends on how it is interpreted by the `dvi` driver.

Virtual fonts come thus in very handy for font manipulation purposes, where new metrics information is simply calculated from that of a base font to obtain stretched, shrunken, letter-spaced and simulated smallcaps (see above) variants. In general, the manipulated metrics must be accompanied by some PostScript code to change the font glyphs themselves. In `dvips` this can be done with code in

the map file, as follows (see Section 22.2.5).

```
ptmr8r Times-Roman "TeXBase1Encoding ReEncodeFont" <8r.enc <timr.pfb
ptmr8rn Times-Roman ".82 ExtendFont TeXBase1Encoding ReEncodeFont" <8r.enc <timr.pfb
ptmro8r Times-Roman ".167 SlantFont TeXBase1Encoding ReEncodeFont" <8r.enc <timr.pfb
```

The first line specifies that the (virtual) font `ptmr8r` is obtained by re-encoding the original to `TeXBase1Encoding` (see Section 21.2.3). Two resources are supplied, the definition of the encoding (`8r.enc`), and the PostScript Type 1 font itself (`timr.pfb`). The second line precedes the above operation with an anamorphical shrinking of the font to 85% of its natural width (`ptmr8rn`), while the third line generates a oblique variant (`ptmro8r`) by slanting the original upright font. The metric file that  $\text{\TeX}$  originally read for these transformed files were created by taking the normal metrics for `Times-Roman` and writing a virtual font with the transformed values. The transformations `ExtendFont`, `ReEncodeFont` and `SlantFont` are defined in PostScript, in the standard header files downloaded by `dvips`. These operations are performed by `dvips` on the PostScript Type 1 fonts after reading the map file while generating the PostScript file (Chapter 22 gives more details on how `dvips` works).

### 21.2.3 $\text{\TeX}$ font encodings

$\text{\TeX}$  itself does not use character names but absolute numeric codes. Hence, a request for typesetting a character is translated into a request to typeset a character at a specific position in a certain font. Moreover, when we want to construct a character using a macro, such as `\' {c}`, we want the effect to be `é`. Therefore  $\text{\TeX}$  has to know where to find this in a font, a nontrivial task if the position of the characters can change from one font to another. There are basically five kinds of font layouts (three for  $\text{\TeX}$ -related and two for PostScript-related fonts):

1. the layout of Knuth's original Computer Modern fonts, which in fact consist of several slightly different layouts, sometimes differing in only one character;
2. the "Cork" extended  $\text{\TeX}$  layout (T1 encoding) (Figure 21.8 on the facing page) plus its sister encodings T2–T7 [15, p. 416];
3. one of the extended ASCII layouts, as used in Windows or Macintosh applications;
4. the Adobe standard layout (the default encoding of PostScript fonts);
5. a re-encoded PostScript layout, such as "TeXBase1" described below (Figure 21.9 on page 28).

This means that the definition of macros like `\'` must depend on the font being referenced or, alternatively, all fonts must use exactly the same font encoding to avoid absolute chaos.  $\text{\TeX}$  implements such commands as "encoding-specific" and provides an interface for specifying different definitions that is activated when the font encoding changes. This even allows the definition of a command like `\'` in one encoding to produce a character composition out of an accent and a base character, and in another encoding to select a single character (See Table 7.33 of [15] for examples).

#### 21.2.3.1 $\text{\TeX}$ 's TeXBase1 encoding

Font names in the PSNFSS packages (see Section 21.3.1) are based on the *Fontname* scheme (Section 21.7) and reflect the encoding used (e.g., the T1-encoded font for *Times Roman* is `ptmr8t`.) On the other hand the `.map` files supplied with PSNFSS (for `dvips`) do *not* use these names but refer to an intermediate "raw" form, which for *Times Roman* is `ptmr8r`. In fact, each font is re-encoded to a new

	'0	'1	'2	'3	'4	'5	'6	'7	
'00x	`	´	^	~	¨	˘	°	˘	"0x
'01x	˘	˘	˘	˘	˘	˘	˘	˘	
'02x	“	”	„	«	»	–	—		"1x
'03x	■	ı	■	ff	fi	fl	ffi	ffl	
'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	[	\	]	^	_	
'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	{		}	~	-	
'20x	Ă	Ą	Ć	Č	Ď	Ě	Ę	Ğ	"8x
'21x	Ł	Ł	Ł	Ń	Ň	■	Ŏ	Ŕ	
'22x	Ř	Ś	Š	Ş	Ť	Ț	Ů	Ű	"9x
'23x	Ÿ	Ž	Ž	Ž	ıı	ı	đ	§	
'24x	ă	ą	ć	č	ď	ě	ę	ğ	"Ax
'25x	ł	ł	ł	ń	ň	■	ŏ	ŕ	
'26x	ř	ś	š	ş	ť	ț	ů	ű	"Bx
'27x	ÿ	ž	ž	ž	ıı	ı	ı	£	
'30x	À	Á	Â	Ã	Ä	Å	Æ	Ç	"Cx
'31x	È	É	Ê	Ë	Ì	Í	Î	Ï	
'32x	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	Ø	"Dx
'33x	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß	
'34x	à	á	â	ã	ä	å	æ	ç	"Ex
'35x	è	é	ê	ë	ì	í	î	ï	
'36x	ð	ñ	ò	ó	ô	õ	ö	œ	"Fx
'37x	ø	ù	ú	û	ü	ý	þ	ß	
	"8	"9	"A	"B	"C	"D	"E	"F	

Figure 21.8: The T1 font layout (*Times Roman*).

encoding standard, TeXBase1, which is used to build T<sub>E</sub>X virtual fonts for various different user encodings, i.e., ptmr7t, ptmr8t, ptmr8y (see the examples at the end of Section 21.7) are virtual fonts referring to the same underlying raw font. Since not all the characters in the Cork encoding are present in a typical PostScript font, it is not possible to simply re-encode the font directly to the final form.

The raw font encoding TeXBase1 was created by an ad hoc group of interested T<sub>E</sub>X font experts, primarily Karl Berry, Berthold Horn, Alan Jeffrey, Pierre MacKay, and Sebastian Rahtz. The aim was to make available for typesetting all the characters normally included in PostScript Type 1 fonts, i.e., the characters in the *Adobe Standard Encoding*, ISO Latin 1, plus a few extra characters available in Lucida Bright.

Figure 21.9 on the following page shows the TeXBase1 layout in a conventional T<sub>E</sub>X font chart, while Figure 21.8 shows the layout corresponding to the T1 encoding. More generally, Tables 21.14

	'0	'1	'2	'3	'4	'5	'6	'7	
'00x		·	fi	fl	/	~	Ł	ł	"0x
'01x	·	°		˘	—		Ž	ž	
'02x	˘	ı							"1x
'03x							`	'	
'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	[	\	]	^	_	
'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	{		}	~		
'20x			,	f	„	...	†	‡	"8x
'21x	^	%	Š	<	Œ				
'22x				“	”	•	—	—	"9x
'23x	~	™	š	>	œ			ÿ	
'24x		ı	¢	£	¤	¥	¦	§	"Ax
'25x	¨	©	ª	«	¬	-	®	-	
'26x	°	±	²	³	´	µ	¶	·	"Bx
'27x	¸	¹	º	»	¼	½	¾	¿	
'30x	À	Á	Â	Ã	Ä	Å	Æ	Ç	"Cx
'31x	È	É	Ê	Ë	Ì	Í	Î	Ï	
'32x	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	×	"Dx
'33x	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß	
'34x	à	á	â	ã	ä	å	æ	ç	"Ex
'35x	è	é	ê	ë	ì	í	î	ï	
'36x	ð	ñ	ò	ó	ô	õ	ö	÷	"Fx
'37x	ø	ù	ú	û	ü	ý	þ	ÿ	
	"8	"9	"A	"B	"C	"D	"E	"F	

Figure 21.9: The TeXBase1 font layout (*Times Roman*).

and 21.15 in Appendix 21.8 contrast these encodings with other commonly used ones.

### 21.2.4 PostScript font encodings

PostScript fonts are arranged as a set of procedures that use a lookup table of *names* for the characters; fonts include a mapping between the names and numbers (the encoding vector), but this can be easily changed. Thus, while the character `exclamdown` (`j`) is *often* mapped to decimal 161, it can always be called by name or mapped to another number.

An encoding vector is a normal PostScript array of 256 elements:

```
/AnEncoding [           % 256 character names follow
  /grave /acute /circumflex /tilde /dieresis /hungarumlaut
```

```

/ring /caron /breve /macron /dotaccent /cedilla
/ogonek /quotesinglbase /guilsinglleft /guilsinglright
. . . . ]

```

When the font is requested, such a vector can be supplied to override the default encoding as specified in the font. No single generally accepted standard for encoding layouts of fonts exists. Only a certain number of characters (essentially the ASCII set), can be found at commonly agreed positions, but these are not sufficient for serious typesetting.

Tables 21.14 and 21.15 in Appendix 21.8 show the normal Roman text characters and their positions in the following widely-used encodings.

EC	the $\text{\TeX}$ T1 encoding, also known as “Cork” (after the $\text{\TeX}$ Users Group meeting in Cork (Ireland) in 1990 where it was agreed upon);
TeXBase1	an arbitrary re-encoding to access all the standard characters in PostScript fonts, used by the PSNFSS virtual fonts (see previous section);
ISO Latin 1 Standard	the extended ASCII set designed to cover western European languages; the default encoding for PostScript fonts, created by Adobe; it does <i>not</i> encode all the characters in a font;
Windows ANSI	a non-standard extended ASCII similar to ISO Latin 1, widely used due to the Microsoft products that support it;
Mac Roman	the Macintosh equivalent of the above, used by all Mac applications;
PDF	another Adobe encoding, the internal one used as standard by the Portable Document Format.

With *current*  $\text{\TeX}$ , PostScript, and fonts two very important facts must be remembered:

1. PostScript font characters are arranged *by name*—there is neither any “right” encoding, nor any overhead in setting up a new encoding;
2. there may exist characters in the font which are *unencoded* by default with the Adobe Standard encoding. A font can even contain more than 256 characters, so that no single encoding can access all of them. In such a case the font can be loaded more than once, with different encodings.

#### 21.2.4.1 Going to 16 bits

The encodings described until now, and the PostScript fonts themselves, are limited to the “eight bit” computer world, which means that each “character” code can only take values between 0 and 255. Modern computer operating systems directly support at least 16 bits and use Unicode [20] as their native encoding. Fonts are also moving to 16 (or even 32) bits, so that 65000 (or more) different characters can be encoded. For the moment, to be used with  $\text{\TeX}$ , such large fonts must still be “subdivided” into subfonts limited to 256 characters (see, e.g., Section 21.6.3, which explains how this is handled with the OpenType font *Minion Pro*.)

Work is ongoing on new extended versions of  $\text{\TeX}$  with direct support of Unicode. Xe $\text{\TeX}$ , (<http://scripts.sil.org/xetex/>) an alternative to the standard  $\text{\TeX}$  (or pdf $\text{\TeX}$ ) typesetting engine, provides enhanced support for Unicode and Mac OS X fonts and several supporting macro packages to simplify the use of Xe $\text{\TeX}$  with existing  $\text{\TeX}$  documents and styles are already available. *Omega* (<http://omega.enstb.org/>) extends  $\text{\TeX}$ ’s 8-bit to 16-bit data-structures for characters and pointers, allows multiple input and output character sets, and uses programmable filters to translate from one encoding to another, to perform contextual analysis, etc. *Omega*, whose prime aim is improving  $\text{\TeX}$ ’s multilingual abilities can easily cope with multiple or complex languages, like Arabic, Indic, Khmer, Chinese, Japanese or Korean, in one document. *Omega* defines a new standard encoding “*TeX Unicode*”, which proposes a typographic implementation of the data exchange Unicode standard. Its

first part (UT1) covers the Latin, IPA, Greek, Cyrillic alphabets and some dingbats. The second part (UT2) covers right-to-left scripts: currently Arabic, Hebrew and Berberian Tifinagh and later on, Syriac. *Omega* will soon be able to do away with T<sub>E</sub>X's `.tfm`, `.vf`, etc. files, and use directly the internal tables of OpenType fonts for typesetting.

### 21.2.5 Types of T<sub>E</sub>X fonts

As explained previously, for typesetting T<sub>E</sub>X needs only the metric information (`.tfm`) for each font. It is up to the `dvi` driver to paint the glyphs on the output medium. Originally a T<sub>E</sub>X-based system could only use fonts created with MetaFont, but thanks to the virtual font mechanism PostScript and other fonts can also be integrated. A short description of font formats associated with the MetaFont-based font technology follows.

#### 21.2.5.1 The `.gf` font format

The `.gf` (generic font) format is produced by MetaFont when generating a font; this is a *bitmap* format, in contrast to the outline format of the MetaFont font source. It is created for a particular device at a particular size (e.g., at 16 points for a Canon laser printer). The `.gf` format does not match the format of any other font software; it was designed to be easy to produce conversion programs to any other format, hence the name “generic”. MetaFont also produces a `.tfm`.

A number of conversion programs are usually part of a T<sub>E</sub>X system, the most important being `gftopk`, which produces a `.pk` file, a format understood by all modern `dvi` driver programs. Other programs are `gftodvi`, which produces large pictures of every character in the font (such as the samples in [9]), intended primarily for test outputs when developing MetaFont fonts, and the program `gftype` which tests whether or not a `.gf` file is corrupted.

#### 21.2.5.2 The `.pk` font format

The normal bitmap format used by `dvi` drivers is the “packed” form (file extension `.pk`). This is simply a more efficient storage of the `.gf` format, and is usually generated using `gftopk` immediately after MetaFont. The size or device characteristics cannot be changed at this stage. It is also possible to use `pkto gf` to translate a `.pk` file back to the generic format, and `pktype` to test its validity.

The user must keep in mind the fact that bitmap fonts cannot be scaled in size without loss of quality, and that MetaFont generates different bitmap patterns for different devices. The very precise tuning that is possible using MetaFont makes for high-quality output, but has the disadvantage that fonts are not portable among different output devices (although in practice many devices, such as 600-dpi laser printers, are similar enough that they can share font files).

### 21.2.6 Types of PostScript fonts

The page description language PostScript can be used to describe complex graphics images in an efficient way. In the case of PostScript fonts, each character can be described by a small PostScript program specifying character *outlines*, i.e., collections of lines, arcs, and curves. Painting the characters on an output device is the task of the PostScript interpreter, which rasterizes these outlines dynamically and transforms them into a bitmap image, taking into account the resolution of the output device (this contrasts with MetaFont, which, as explained in Section 21.2.5, performs the rasterization once and for all at the stage at which the `.gf` file is created).

The following types of PostScript fonts exists [2, Chapter 5]:

**Type 0** A composite font, i.e., a font composed of other fonts, organized hierarchically. Composite



font programs can contain several thousand characters, accessed by multi-byte codes. They are mostly useful for non-Roman scripts, such as Kanji characters, which can also have several writing directions. The font program can contain several sets of metrics, and a key in the font dictionary selects the set of metrics used to show a string.

**Type 1** A font that defines glyph shapes by using a special encoded format. An extension is the multiple-master font format, which allows the generation of a wide variety of typeface styles from a single font. Both of these will be briefly discussed in the following.

**Type 2** A Compact Font Format (CFF) font. Together with the Type 14 (Chameleon) font format it provides a compact representation that enables multiple fonts to be stored as a unit called a “font set”. Since CFF is based on the Type 1 technology it retains full fidelity to the original fonts, while achieving significant space reduction due to a compact binary representation and sharing of data that is common to multiple fonts.

**Type 3** A font format that defines glyphs with ordinary PostScript procedures. It is briefly described later.

**Types 9, 10, 11, 32** These font types are used with CID-keyed fonts. CID-keyed fonts provide a convenient and efficient method for defining multiple-byte character encodings, base fonts with a large number of glyphs, and composite fonts that use these base fonts and character encodings. Additionally, they provide straightforward methods for creating a rearranged font, which selects glyphs from one or more existing fonts by means of a revised encoding. These capabilities provide great flexibility for representing text in writing systems for languages with large character sets, such as Chinese, Japanese, and Korean.

**Type 14** The Chameleon font format. It implements a “shape library”, which allows a compact representations of Latin-text fonts. It consists of a master font and its font descriptor database: the master font is tailored to address the needs of a particular product, while the font descriptors define how to extract fonts of interest from the master. Typically, there is one font set for all CFF fonts and one each for a Chameleon master font and its descriptor database.

**Type 42** A font wrapper for the TrueType font format.

We now briefly describe the Type 1, Type 3, and Multiple Master formats, which are the more relevant for use with  $\TeX$ .

### 21.2.6.1 Type 1 fonts

The Type 1 font format (see also Sections 21.1.1 and 21.1.2) provides a compact way of describing a font outline using a subset of the PostScript language optimized for efficiency at runtime. Since each character is described as an outline format its rendering has unlimited resolution. A PostScript Type 1 font is organized as a collection of named procedures describing character shapes. When a character code is requested, the Type 1 renderer first uses the character code as an entry in an *encoding* array in the font dictionary to obtain the name of the character to be constructed.

A Type 1 font is characterized by its font name (`FullName`) and `Encoding`, which usually is Adobe’s “`StandardEncoding`”, but some fonts (such as expert variants) may have specific encodings.

Type 1 fonts are typically supplied by a vendor in a form suitable for a Macintosh, Microsoft Windows or Unix system. In particular, the Font Binary form (`.pfb`) is used on Unix. One can convert Type 1 files from one format into another with the help of the `t1utils` programs.<sup>1</sup>

<sup>1</sup>A collection of type 1 font manipulation programs (<http://www.lcdf.org/type/#t1utils>), originally written by Lee Hetherington and revised by Eddie Kohler. `t1ascii` changes `.pfb` (binary) fonts into `.pfa` (ASCII) format; `t1binary` does the reverse. `t1disasm` translates the `.pfb` or `.pfa` format into a human-readable and editable form, while `t1asm` does the

### 21.2.6.2 Type 3 fonts

The PostScript Type 3 font format has long been a favourite technology for many people because of the flexibility it offers, since the entire PostScript language can be used to describe character shapes. This means that Type 3 fonts can have more elaborate designs than Type 1 fonts: the glyphs can contain shades of gray, graduated fills or variable stroke widths. Moreover, it is easier to create a Type 3 font than to use the PostScript Type 1 technology.

On the other hand, it is impossible to use hints, and character rendering is less efficient (slower) than for Type 1 fonts. Historically, PostScript `dvi` drivers often used the Type 3 format to include output from MetaFont, which generates bitmapped fonts from outline source descriptions. The `dvi` driver extracts bitmaps from the packed font images (`.pk` files) for all  $\TeX$  fonts used in a  $\TeX$  job, and transforms them into Type 3 PostScript fonts for inclusion in the output. However, these fonts only look well at the resolution for which they were originally generated, and at high resolutions such `.pk` files become huge in size. Therefore, at present Type 1 renderings of the  $\TeX$  fonts should be preferred.

### 21.2.6.3 Multiple Master fonts

The PostScript Type 1 multiple master (MM) format is an extension of the Adobe PostScript Type 1 font format. Essentially, it allows two (or more) design variations to be encoded on a given design axis. Afterwards, any in-between state (*instance*) may be generated by the user as required. For instance, a MM font featuring a *weight* axis could have an ultra-light master and an extra-black master, thus allowing any conceivable variation in between. Multiple axes are also possible, with each additional axis doubling the number of master fonts required, since each possible extreme must be designed separately. Figure 21.10 shows the design space of a three-axis MM font in which the axes are *weight*, *width*, and *optical size*, characterized by the three master designs specified at its corners.

Figure 21.11 shows some custom-made font instances generated with the two-axis Myriad MM font, while Figure 21.12 shows similar font instances generated with the three-axis Minion MM font.

Since 2000 Adobe started phasing out its multiple master fonts since most applications cannot handle them properly and, from an economic and marketing standpoint, it make more sense to release fontsets with multiple separate OpenType fonts. In fact, the *Minion Pro* OpenType font that we describe in Section 21.6.3, is part of one of the four font sets that replace (and extend) the Minion MM variant.

### 21.2.7 Making all those files work together

One of the commonest sources of confusion for would-be PostScript font users is knowing which files they have to install and what they have to change in their driver setup. Here we describe in more detail what files are involved in the PSNFSS setup, in particular when using the `dvips` driver.<sup>1</sup>

The PSNFSS packages are implemented using *virtual fonts* (see Section 21.2.2); this means that for every font that is used, there are three font files and an entry in a font description file. Let us consider the common case of Times Bold. With the PSNFSS setup the package file `times.sty` looks something like this:

```
\ProvidesPackage{times}[2004/09/15 PSNFSS-v9.2 (SPQR)]
\renewcommand{\sfdefault}{phv}
\renewcommand{\rmdefault}{ptm}
\renewcommand{\ttdefault}{pcr}
```

When  $\TeX$  typesets something in Roman, it looks for a *font-encoding ffile* that is named by combining the encoding name and the family name defined by `\rmdefault`. The default encoding is `T1` and the

reverse, `t1unmac` translates a Macintosh Type 1 font into either `.pfb` or `.pfa` format, `t1mac` does the reverse.

<sup>1</sup>Other `dvi` drivers follow a similar approach, please consult their documentation.

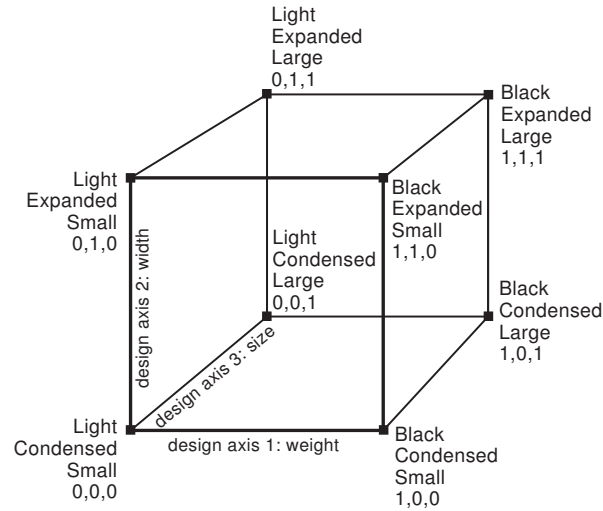


Figure 21.10: Multiple Master typeface design space.

Hxkp Hxkp Hxkp Hxkp Hxkp Hxkp Hxkp  
Hxkp Hxkp Hxkp Hxkp Hxkp Hxkp Hxkp  
Hxkp Hxkp Hxkp Hxkp Hxkp Hxkp Hxkp  
Hxkp Hxkp Hxkp Hxkp Hxkp Hxkp Hxkp  
Hxkp Hxkp Hxkp Hxkp Hxkp Hxkp Hxkp

Figure 21.11: The two-axis Multiple Master Myriad sans serif font. Weights increase from left to right and widths from top to bottom.

Hxkp Hxkp Hxkp Hxkp Hxkp Hxkp  
Hxkp Hxkp Hxkp Hxkp Hxkp Hxkp  
Hxkp Hxkp Hxkp Hxkp Hxkp Hxkp  
Hxkp Hxkp Hxkp Hxkp Hxkp Hxkp  
Hxkp Hxkp Hxkp Hxkp Hxkp Hxkp  
Hxkp Hxkp Hxkp Hxkp Hxkp Hxkp

Figure 21.12: The three-axis Multiple Master Minion serif font. The top line shows various optical sizes (6pt, 8pt, 11pt, 18pt, 40pt, and 72pt) normalized to 20pt. The bottom matrix shows various weights (increasing from left to right) and widths (increasing from top to bottom).

package defines `\rmdefault` to be `ptm`, so  $\text{\TeX}$  opens `t1ptm.fd`,<sup>1</sup> which looks like this:

```
\ProvidesFile{t1ptm.fd}[2001/06/04 font definitions for T1/ptm.]
\DeclareFontFamily{T1}{ptm}{}

\DeclareFontShape{T1}{ptm}{m}{n}{<-> ptmr8t}{}
\DeclareFontShape{T1}{ptm}{m}{sc}{<-> ptmrc8t}{}
\DeclareFontShape{T1}{ptm}{m}{sl}{<-> ptmro8t}{}
\DeclareFontShape{T1}{ptm}{m}{it}{<-> ptmri8t}{}
\DeclareFontShape{T1}{ptm}{b}{n}{<-> ptmb8t}{}
\DeclareFontShape{T1}{ptm}{b}{sc}{<-> ptmbc8t}{}
\DeclareFontShape{T1}{ptm}{b}{sl}{<-> ptmbo8t}{}
\DeclareFontShape{T1}{ptm}{b}{it}{<-> ptmbi8t}{}

\DeclareFontShape{T1}{ptm}{bx}{n}{<->ssub * ptm/b/n}{}
\DeclareFontShape{T1}{ptm}{bx}{sc}{<->ssub * ptm/b/sc}{}
\DeclareFontShape{T1}{ptm}{bx}{sl}{<->ssub * ptm/b/sl}{}
\DeclareFontShape{T1}{ptm}{bx}{it}{<->ssub * ptm/b/it}{}

```

This file provides a font name for each allowed combination of encoding, family, series, and shape. If we are setting text in bold, the default series used by e.g., `\textbf` is `bx`; assuming we want the normal shape,  $\text{\TeX}$  can extract a line from the above to match encoding `T1`, family `ptm`, series `bx` and shape `n`. This yields `<->ssub * ptm/b/n`, which means that ordinary bold is silently substituted for “bold extended”. A second search for `T1 + ptm + b + n` gives us `<-> ptmb8t`. The `<->` means that we use the same font for all sizes, appropriately scaled, and the font name is `ptmb8t`. Thus the basic  $\text{\TeX}$  engine now loads `ptmb8t.tfm` (if it can find it) and the job proceeds.

$\text{\TeX}$ 's job is now done, and we turn to the `.dvi` file, which contains a request for font `ptmb8t`. The device driver tries to satisfy this demand by attempting the following sequence of actions:

1. load a *virtual font* of the right name that gives further instructions;
2. check whether the font is built into the printer, and thus does not need to be found;
3. check whether the font can be used directly in the printer but needs to be downloaded;
4. find a bitmap `.pk` font file.

The virtual font `ptmb8t` consists of a set of instructions for each character; most of these simply tell the driver to fetch the appropriate character from another, “raw,” font `ptmb8r` (see Section 21.2.3 for more on the mechanics of virtual fonts). Thus the driver starts looking for `ptmb8r`; in theory this could go on through several cycles of virtual fonts, but in practice the driver locates this name in its list of built-in fonts. How this is done depends on the driver; Section 22.2.5 discusses Tom Rokicki's `dvips`, which uses `.map` files to specify the built-in fonts, but other drivers have a similar concept. The `.map` files cannot be used directly with drivers other than `dvips` but they contain the information needed to integrate the font. For example, the line

```
ptmb8r Times-Bold "TeXBase1Encoding ReEncodeFont" <8r.enc
```

means that the  $\text{\TeX}$  font name `ptmb8r` corresponds to the PostScript font `Times-Bold` after re-encoding it using the `TeXBase1Encoding` (which is stored in the file `8r.enc` that needs to be downloaded to the printer). For details on interpreting the `.map` files for `dvips`, see Section 22.2.5.

To summarize (Figure 21.13), for PSNFSS for every font family the following resources must be installed:

<sup>1</sup>The encoding name is translated to lower-case before the file is looked for.

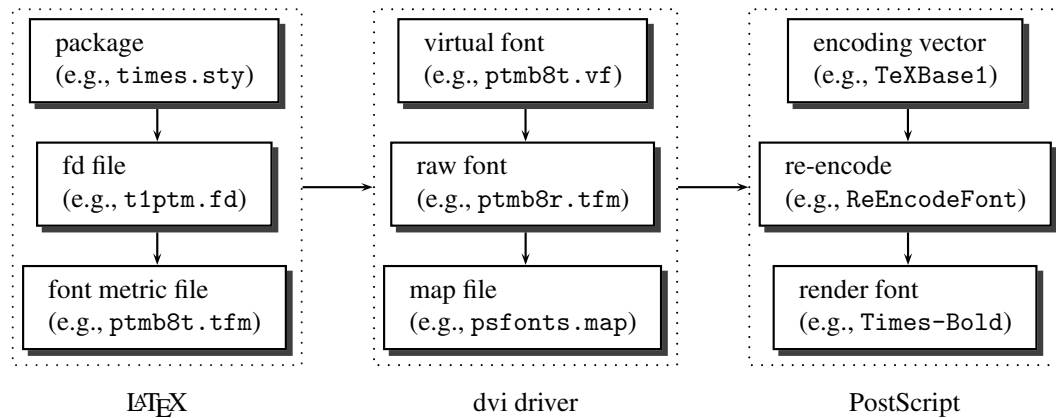


Figure 21.13: Files and processes used by  $\text{\LaTeX}$ , a dvi-to-PostScript driver and PostScript.

1. font description .fd files for each supported encoding for the font family (e.g., ot1ptm.fd and t1ptm.fd);
2. font metric .tfm files for each font in the family in each encoding (e.g., ptmr7t.tfm, ptmr8t.tfm, ptmb7t.tfm, ptmb8t.tfm, etc.);
3. virtual font .vf files for each metric file (e.g., ptmr7t.vf, ptmr8t.vf, ptmb7t.vf, ptmb8t.vf, etc.);
4. font metric .tfm files for each of the final font names that emerge from the virtual font process (e.g., ptmr8r.tfm, ptmb8r.tfm, etc.)—note that ptmr7t and ptmr8t resolve into references to a single .tfm file, ptmr8r;
5. entries in a driver map file defining the correspondence between the  $\text{\TeX}$  font names (such as ptmr8r), the PostScript fonts, and possibly re-encodings of such fonts;
6. files containing the PostScript encoding vectors referred to in the map file (e.g., 8r.enc);
7. and (possibly) the fonts themselves to be downloaded.

By building on the virtual font mechanism PSNFSS offers vastly more flexibility than simple re-encoding and provides a clean layer of functionality between  $\text{\TeX}$  and PostScript. Moreover, PSNFSS does not use the native font name, *Times-Roman*, in  $\text{\TeX}$  because font names are not standardized and a rational naming scheme (see Section 21.7) helps a great deal in managing the large quantity of available fonts.

## 21.3 PSNFSS: using freely available PostScript Type 1 fonts

If you want to use PostScript fonts, three situations are possible:

1. You are content to use the standard setup distributed with the  $\text{\LaTeX}$  PSNFSS package, discussed below (Section 21.3.1).
2. The metrics and description files have already been created for the font family you want, so you only have to install these  $\text{\LaTeX}$  support files, and get and install the actual fonts; this is discussed in Sections 21.3.3 and 21.4.

<i>Package</i>	<i>Roman Font</i>	<i>Sans Serif Font</i>	<i>Typewriter Font</i>	<i>Formulas</i>
(none)	CM Roman	CM Sans Serif	CM Typewriter	CM Math
mathptmx	Times			Times + Symbol
mathpazo	Palatino			Palatino + Pazo
charter	Charter**			
utopia*	Utopia**			
chancery	Zapf Chancery			
helvet		Helvetica		
avant		Avant Garde		
courier			Courier	
bookman	Bookman	Avant Garde	Courier	
newcent	New Century Schoolbook	Avant Garde	Courier	
<i>Obsolete Packages</i>				
times	Times	Helvetica	Courier	
palatino	Palatino	Helvetica	Courier	
mathptm	Times			Times + Symbol + CM
mathpple	Palatino			Palatino + Symbol + Euler

\* An alternative package that includes math support is `fourier` (CTAN: `fourier`, see also [15, Section 7.7.7]).

\*\* These fonts might have to be downloaded separately, see Section 21.3.3.

Table 21.1: Fonts used by PSNFSS packages

3. You have only the Adobe Font Metric files and need to create all the metric files etc. that  $\text{\TeX}$  needs; this is rather more complicated and is not described here.<sup>1</sup>

### 21.3.1 The standard PSNFSS system

The PSNFSS bundle, originally developed by Sebastian Rahtz, offers a complete working set-up of the  $\text{\TeX}$  font selection scheme for use with common PostScript fonts, covering the “Base 35” fonts (which are built into any Level 2 PostScript printing device and the ghostscript interpreter) and the free Charter and Utopia fonts.<sup>2</sup> The current implementation of PSNFSS is maintained by Walter Schmidt and is part of the required set of support files for  $\text{\TeX}$  that should be available with every  $\text{\TeX}$  installation.

For normal use you will probably have to include only one (or more) of the packages listed in Table 21.1 to change the default Roman, sans serif, and/or typewriter typefaces.

Most people simply want to install support for the fonts available in almost all PostScript printers. These are four serif families (Times, Palatino, Bookman, and New Century Schoolbook), two sans serif families (Helvetica and AvantGarde), one monospaced typewriter family (Courier), a symbol font, and the cursive Zapf Chancery; these were the fonts Apple provided with the first LaserWriter Plus in 1986. Together these families, in their various shapes, make up 35 fonts, and you often see references to “the 35 fonts” (see Table 21.2 for a sample). If you stick to these fonts, you can be sure that everyone can print your documents.

<sup>1</sup>An excellent step-by-step guide is Philipp Lehman’s *Font installation guide*, available at CTAN:info/Typelfonts/fontinstallationguide/fontinstallationguide.pdf. You should also have a good knowledge of  $\text{\TeX}$ ’s font-selection scheme, as documented in Chapter 7 of [15]; a condensed explanation can be found in the file `fontguide.tex`, which is part of the  $\text{\TeX}$  distribution.

<sup>2</sup>If these fonts are missing from your  $\text{\TeX}$  installation read Section 21.3.3.

Exa.  
21-3-1

**Times Roman** A floating field for £45. ¡THE DAZED BROWN FOX HAD 12345–67890 JUMPS! — ¿Are Kafka’s Schloß and Æsop’s Œuvres naïve vis-à-vis a dæmonic phœnix’s rôle and its soufflés?

**Palatino** A floating field for £45. ¡THE DAZED BROWN FOX HAD 12345–67890 JUMPS! — ¿Are Kafka’s Schloß and Æsop’s Œuvres naïve vis-à-vis a dæmonic phœnix’s rôle and its soufflés?

**New Century Schoolbook** A floating field for £45. ¡THE DAZED BROWN FOX HAD 12345–67890 JUMPS! — ¿Are Kafka’s Schloß and Æsop’s Œuvres naïve vis-à-vis a dæmonic phœnix’s rôle and its soufflés?

**AvantGarde** A floating field for £45. ¡THE DAZED BROWN FOX HAD 12345–67890 JUMPS! — ¿Are Kafka’s Schloß and Æsop’s Œuvres naïve vis-à-vis a dæmonic phœnix’s rôle and its soufflés?

**Bookman** A floating field for £45. ¡THE DAZED BROWN FOX HAD 12345–67890 JUMPS! — ¿Are Kafka’s Schloß and Æsop’s Œuvres naïve vis-à-vis a dæmonic phœnix’s rôle and its soufflés?

**Helvetica** A floating field for £45. ¡THE DAZED BROWN FOX HAD 12345–67890 JUMPS! — ¿Are Kafka’s Schloß and Æsop’s Œuvres naïve vis-à-vis a dæmonic phœnix’s rôle and its soufflés?

**Courier** A floating field for £45. ¡THE DAZED BROWN FOX HAD 12345–67890 JUMPS! -- ¿Are Kafka’s Schloß and Æsop’s Œuvres naïve vis-à-vis a dæmonic phœnix’s rôle and its soufflés?

**Zapf Chancery** *A floating field for £45. ¡THE DAZED BROWN FOX HAD 12345–67890 JUMPS! — ¿Are Kafka’s Schloß and Æsop’s Œuvres naïve vis-à-vis a dæmonic phœnix’s rôle and its soufflés?*

Table 21.2: Sample texts for standard PostScript fonts.

Usually T<sub>E</sub>X installation have PSNFSS pre-installed. If this is not the case, you should obtain the PSNFSS collection (CTAN:macros/latex/required/psnfss/). Read the file 00readme.txt in that directory to install all relevant files. The PSNFSS packages redefine the default roman, sans serif or typewriter family, e.g., the bookman package consists of:

```
\renewcommand{\rmdefault}{pbk}
\renewcommand{\sfdefault}{pag}
\renewcommand{\ttdefault}{pcr}
```

The only difficulty is knowing the names of the font families. In fact the PSNFSS bundle uses the Karl Berry naming scheme, which is discussed in Section 21.7.

In Table 21.1 on the preceding page only two packages attempt to set up new fonts for math and the first eight packages only change fonts in one of the three text font categories. Thus, to get Times as the Roman text font, Helvetica as the sans serif text font, and Courier as the typewriter text font, one would need to load mathptmx, helvet, and courier. So why is it that the times package, which does this all in one go, is considered obsolete?

One reason is that Helvetica, if loaded at its nominal size, is actually too large to blend well with Times or Courier. That does not too important is you use Helvetica only for headings. But if Times and Helvetica are going to be mixed in running text (something that is made easy by L<sup>A</sup>T<sub>E</sub>X commands such as \textsf), then using a package such as times will produce questionable results. The helvet package, on the other hand, offers the ability to scale the fonts by specifying the option scaled, which scales the fonts down to 95% of the requested size. This option is actually a keyword/value option, so that even



finer control is possible—`scaled=0.92` would load the fonts at 92% of their nominal size.

However, if you do not want to change the  $\text{\TeX}$ 's default math font set-up, it might anyway be useful to load the times package. In that case you can load the helvet package afterwards to apply scaling.

The PSNFSS collection contains only two packages that modify the math set-up: `mathptmx` selects math fonts that blend with Times Roman and `mathpazo` selects math fonts designed to work with Palatino (they are described in Sections 7.6.2 and 7.6.3 of *κ e L<sup>A</sup>T<sub>E</sub>X Companion, 2ed* [15]). Several non-PSNFSS packages are available for typesetting math, some based on free math fonts while others use commercial fonts.<sup>1</sup>

Sans serif as document  
typeface

By default,  $\text{\TeX}$  selects a Roman typeface as the document font. Packages like `helvet` or `avant` change the default sans serif typeface (by changing `\sfdefault`) but do not change the default document font family. If such a typeface should be used as the document font the following line should be added to the preamble of your document:

```
\renewcommand\familydefault{\sfdefault}
```

The PSNFSS collection not only provides support for the common PostScript text fonts, but it also contains the `pifont` package, which is described next.

### 21.3.2 pifont—Accessing Pi and Symbol fonts

Fonts containing collections of special symbols, which are normally not found in a text font, are called “Pi fonts.” One such font, the PostScript font Zapf Dingbats, is available if you use the `pifont` package, originally written by Sebastian Rahtz, and now part of PSNFSS.

Accessing glyphs from  
Zapf Dingbats

The directly accessible characters of the PostScript Zapf Dingbats font are shown in Table 21.3 on the facing page. A given character can be chosen via the `\ding` command. The parameter for the `\ding` command is an integer that specifies the character to be typeset according to the table. For example, `\ding{38}` gives ☺.

The `dinglist` environment is a variation of the `itemize` list. The argument specifies the number of the character to be used at the beginning of each item.

➤ The first list item.	<code>\usepackage{pifont}</code>
➤ The second list item.	<code>\begin{dinglist}{253}</code>
➤ The third list item.	<code>\item The first list item. \item The second list item. \item The third list item.</code>
	<code>\end{dinglist}</code>

Exa.  
21-3-2

The environment `dingautolist` allows you to build an enumerated list from a sequence of Zapf Dingbats characters. The argument of the environment specifies the number of the first character of the sequence. Subsequent items will be numbered by incrementing this number by one. This makes some starting positions like 172, 182, 192, and 202 in Table 21.3 on the next page very attractive, as

<sup>1</sup>See, for instance, [15, Section 8.8.3], [7, Chapter 10], or Stephen G. Hartke's article *A survey of Free Math Fonts for T<sub>E</sub>X and L<sup>A</sup>T<sub>E</sub>X*, available at <http://tug.org/pracjourn/2006-1/hartke/hartke.pdf>.



40	✈	41	☒	42	☞	43	☛	44	✂	45	✂	46	✂	47	☞	48	☞	49	☞
50	✂	51	✓	52	✓	53	✕	54	✕	55	✕	56	✕	57	✕	58	✕	59	✕
60	✕	61	†	62	†	63	†	64	✕	65	✕	66	✕	67	✕	68	✕	69	✕
70	✕	71	✕	72	★	73	☆	74	✕	75	☆	76	★	77	★	78	★	79	★
80	✕	81	✕	82	✕	83	✕	84	✕	85	✕	86	✕	87	✕	88	✕	89	✕
90	✕	91	✕	92	✕	93	✕	94	✕	95	✕	96	✕	97	✕	98	✕	99	✕
100	✕	101	✕	102	✕	103	✕	104	✕	105	✕	106	✕	107	✕	108	●	109	○
110	■	111	□	112	□	113	□	114	□	115	▲	116	▼	117	◆	118	◆	119	◆
120		121		122		123	•	124	•	125	•	126	•	127	•	128	•	129	•
130	•	131	•	132	•	133	•	134	•	135	•	136	•	137	•	138	•	139	•
140	•	141	•	142	•	143	•	144	•	145	•	146	•	147	•	148	•	149	•
150	•	151	•	152	•	153	•	154	•	155	•	156	•	157	•	158	•	159	•
160	•	161	•	162	•	163	•	164	•	165	•	166	•	167	•	168	•	169	•
170	♥	171	♠	172	①	173	②	174	③	175	④	176	⑤	177	⑥	178	⑦	179	⑧
180	⑨	181	⑩	182	⑪	183	⑫	184	⑬	185	⑭	186	⑮	187	⑯	188	⑰	189	⑱
190	⑲	191	⑳	192	㉑	193	㉒	194	㉓	195	㉔	196	㉕	197	㉖	198	㉗	199	㉘
200	㉙	201	㉚	202	㉛	203	㉜	204	㉝	205	㉞	206	㉟	207	㊱	208	㊲	209	㊳
210	㊴	211	㊵	212	➔	213	➔	214	↔	215	↕	216	➔	217	➔	218	➔	219	➔
220	➔	221	➔	222	➔	223	➔	224	➔	225	➔	226	➔	227	➔	228	➔	229	➔
230	➔	231	➔	232	➔	233	➔	234	➔	235	➔	236	➔	237	➔	238	➔	239	➔
240	➔	241	➔	242	➔	243	➔	244	➔	245	➔	246	➔	247	➔	248	➔	249	➔
250	➔	251	➔	252	➔	253	➔	254	➔										

Table 21.3: The characters in the PostScript font ZapfDingbats

40	(	41	)	42	*	43	+	44	,	45	-	46	.	47	/	48	0	49	1
50	2	51	3	52	4	53	5	54	6	55	7	56	8	57	9	58	:	59	;
60	<	61	=	62	>	63	?	64	≡	65	A	66	B	67	X	68	Δ	69	E
70	Φ	71	Γ	72	H	73	I	74	ϑ	75	K	76	Λ	77	M	78	N	79	O
80	Π	81	Θ	82	P	83	Σ	84	T	85	Y	86	ζ	87	Ω	88	Ξ	89	Ψ
90	Z	91	[	92	∴	93	]	94	⊥	95	—	96	—	97	α	98	β	99	χ
100	δ	101	ε	102	φ	103	γ	104	η	105	ι	106	φ	107	κ	108	λ	109	μ
110	v	111	o	112	π	113	θ	114	ρ	115	σ	116	τ	117	υ	118	ω	119	ω
120	ξ	121	ψ	122	ζ	123	{	124		125	}	126	~	127	~	128	~	129	~
130	~	131	~	132	~	133	~	134	~	135	~	136	~	137	~	138	~	139	~
140	~	141	~	142	~	143	~	144	~	145	~	146	~	147	~	148	~	149	~
150	~	151	~	152	~	153	~	154	~	155	~	156	~	157	~	158	~	159	~
160	~	161	~	162	~	163	~	164	~	165	~	166	~	167	~	168	~	169	~
170	♠	171	↔	172	←	173	↑	174	→	175	↓	176	°	177	±	178	″	179	≥
180	×	181	∞	182	∂	183	•	184	÷	185	≠	186	≡	187	≈	188	...	189	
190	—	191	↙	192	↘	193	↗	194	↖	195	↗	196	⊗	197	⊕	198	⊗	199	∩
200	∪	201	⊃	202	⊇	203	⊂	204	⊆	205	⊆	206	∈	207	∉	208	∠	209	∇
210	®	211	©	212	™	213	Π	214	√	215	·	216	¬	217	∧	218	∨	219	↔
220	←	221	↑	222	⇒	223	↓	224	◇	225	∠	226	®	227	©	228	™	229	Σ
230	∫	231		232	∫	233	∫	234		235	∫	236	∫	237	∫	238	∫	239	
240	∫	241	∫	242	∫	243	∫	244		245	∫	246	∫	247		248	∫	249	
250		251		252		253		254											

Table 21.4: Glyphs in the PostScript font Symbol

differently designed circled number sequences (1–10) start there.

❶ The first item in the list.

❷ The second item in the list.

❸ The third item in the list.

References to list items work as expected: ❶,  
❷, ❸

```
\usepackage{pifont}
\begin{dingautolist}{202}
\item The first item in the list.\label{lst:a}
\item The second item in the list.\label{lst:b}
\item The third item in the list.\label{lst:c}
\end{dingautolist}
References to list items work as expected:
\ref{lst:a}, \ref{lst:b}, \ref{lst:c}
```

Exa.  
21-3-3

The command `\dingline` fills a complete line (leaving 0.5 inch space at left and right) with a given character specified as its argument. For filling parts of a line, use the command `\dingfill`. It acts like  $\TeX$ 's `\dotfill` command, but uses the specified glyph instead of dots.

```
\usepackage{pifont}
\dingline{34} \par\medskip
\noindent\dingfill{233} text text
\dingfill{235} text text \dingfill{236}
```

⇒ ⇒ ⇒ text text ⇒ ⇒ text text ⇒ ⇒ ⇒

Exa.  
21-3-4

Besides providing direct support for the Zapf Dingbats font, the `pifont` package includes a general mechanism for coping with any Pi font that conforms to the NFSS classification  $U/family/m/n$ —for example, the Symbol font with the family name `psy`, or the *Minion Pro* font (see Table 21.7 on page 65).

Accessing individual  
glyphs from a Pi font

To access individual glyphs from such a Pi font, use the `\Pisymbol` command, which takes the *family* name as its first argument and the glyph position in the font as its second argument. Using this command one can readily access the characters in the Symbol font, shown in Table 21.4 on the previous page. For example, `\Pisymbol{psy}{210}` gives ®. In fact, `\ding` (discussed previously) is simply an abbreviation for `\Pisymbol` with the first argument set to `pzd`.

You can also make itemized lists using `Pilist` or enumerated lists using the `Piautolist` environments as follows (we take characters from both the Zapf Dingbats as well as the SYmbol font):

❶ level 1 item 1

α level 2 item 1

β level 2 item 2

✕ level 3 item 1

⊕ level 3 item 2

χ level 2 item 3

❷ level 1 item 2

A reference to level 3 item 1 ✕.

```
\usepackage{pifont}
\begin{Piautolist}{pzd}{182}
\item level 1 item 1
\begin{Piautolist}{psy}{97}
\item level 2 item 1 \item level 2 item 2
\begin{Piautolist}{pzd}{56}
\item level 3 item 1 \label{pilabel}
\item level 3 item 2
\end{Piautolist}
\item level 2 item 3
\end{Piautolist}
\item level 1 item 2
\end{Piautolist}
A reference to level 3 item 1\ref{pilabel}.
```

Exa.  
21-3-5

Similarly, the `\dingline` and `\dingfill` commands are abbreviations for the more general commands `\Piline` and `\Pifill`, as shown below. The example reveals curious gaps in the last line. They are due to `\Piline` and `\Pifill` typesetting their symbols on an invisible grid so that

symbols on different lines come out vertically aligned.

Exa.  
21-3-6

⇒ ⇒ ⇒ text ⇔ ⇔ ⇔ text ⇐ ⇐ ⇐

```
\usepackage{pifont}
\Piline{pzd}{36}      \par\medskip
\noindent\Pifill{psy}{222} text
\Pifill{psy}{219}text\Pifill{psy}{220}
```

### 21.3.3 Installing Charter and Utopia

The simplest case for installing additional font files is complementing the PSNFSS setup with Adobe's *utopia* or Bitstream's *charter* PostScript Type 1 files from CTAN (they are not free software, but the .pfb and .afm files are available as CTAN:fonts/charter.zip and CTAN:fonts/utopia.zip). Once you have downloaded the fonts you can proceed as explained next.

We first install Utopia on Linux with a T<sub>E</sub>X Live setup.

```
[1]> unzip utopia.zip
[2]> cd utopia
[3]> ls
putb8a.afm putbi8a.afm putr8a.afm putri8a.afm readme.utopia
putb8a.pfb putbi8a.pfb putr8a.pfb putri8a.pfb
[4]> mkdir -p /TL2005/texmf-local/fonts/type1/public/adobe/utopia
[5]> cp *.pfb /TL2005/texmf-local/fonts/type1/public/adobe/utopia
[6]> texhash
[7]> updmap --enable Map utopia.map
[8]> latex fonttest
./fonttest.tex
LaTeX2e <2003/12/01>
... a few lines deleted
(/TL2005/texmf-dist/tex/latex/fourier/utopia.sty)
No file fonttest.aux.
(/TL2005/texmf-dist/tex/latex/psnfss/tlput.fd)
[1] (./fonttest.aux) )
Output written on fonttest.dvi (1 page, 796 bytes).
Transcript written on fonttest.log.
[9]> dvips fonttest
This is dvips(k) 5.95b Copyright 2005 Radical Eye Software (www.radicleye.com)
' TeX output 2006.03.19:1509' -> fonttest.ps
<tex.pro><8r.enc><texps.pro>. <putbi8a.pfb><putb8a.pfb><putri8a.pfb>
<putr8a.pfb>[1]
```

After unzipping (step [1]) the .pfb files are created in the *utopia* subdirectory, to which we proceed and list the files (steps [2] and [3]). We see four .pfb (the PostScript Type 1 sources) and .afm (the Adobe metrics) files. In the local tree (texmf-local we then create a subdirectory *adobe/utopia* under *fonts/type1/public/* (step [4]) and copy the PostScript Type 1 .pfb files to that directory (step [5]). We have to update the T<sub>E</sub>X files database to inform it where we installed the files (step [6]) and then add the *utopia.map* file to the list of all font map files available to dvips and pdfT<sub>E</sub>X (step [7]). We finally test the setup by running L<sup>A</sup>T<sub>E</sub>X on *fonttest.tex* which typesets a few font variants of the uppercase and lowercase alphabet (step [8]). It is seen that L<sup>A</sup>T<sub>E</sub>X loads the *utopia* package (in fact *fourier* could be used instead, as suggested in Table *vreftab:psnfssstyles*) and the *tlput.fd* font definition file for OPTima. Finally, dvips run on the dvi file (step [9]) generates the result shown in Figure 21.14 on the following page. Notice how dvips loads the four .pfb files that we copied into the T<sub>E</sub>X tree in step [5].

On a system where Utopia is installed it is easy to access the font without actually loading any package, e.g., via a call to `\usefont`. Moreover, as a PostScript Type 1 font can be scaled to any size

no: abcdefghijklmnopqrstuvwxyz (1234567890)  
 ABCDEFGHIJKLMNOPQRSTUVWXYZ  
 SC: ABCDEFGHIJKLMNOPQRSTUVWXYZ  
*it: abcdefghijklmnopqrstuvwxyz (1234567890)*  
 ABCDEFGHIJKLMNOPQRSTUVWXYZ  
**bd: abcdefghijklmnopqrstuvwxyz (1234567890)**  
 ABCDEFGHIJKLMNOPQRSTUVWXYZ  
*bi: abcdefghijklmnopqrstuvwxyz (1234567890)*  
 ABCDEFGHIJKLMNOPQRSTUVWXYZ

Figure 21.14: Adobe Utopia font sample

with the `\fontsize` command, we can combine the two to typeset headings or title pages at font sizes not predefined by  $\TeX$ .

*Utopia-Italic*

```
\fontsize{12mm}{14mm}% select size
\usefont{T1}{put}{m}{it}% select font
\centering Utopia-Italic
```

Exa.  
21-3-7

On Microsoft Windows with Mik $\TeX$  we shall now go through the same procedure and show how to install the Bitstream *Charter* font. We will find that the various steps are more or less equivalent.

```
[1]> cd charter
[2]> mkdir \TL2005\localtexmf\fonts\type1\public\bitstrea\charter
[3]> copy *.pfb \TL2005\localtexmf\fonts\type1\public\bitstrea\charter
bchb8a.pfb
bchbi8a.pfb
bchr8a.pfb
bchri8a.pfb
      4 file(s) copied.
[4]> texhash
Creating the file name database...
Deleting "C:\TL2005\localtexmf\miktex\config\texmf0.fndb"...
C:\TL2005\localtexmf[dvipdfm][dvips][fonts][miktex][pdftex] done
Deleting "C:\TL2005\localtexmf\miktex\config\texmf1.fndb"...
C:\TL2005\texmf[bibtex][doc][dvipdfm][dvips][etex][fontname][fonts]
[ghostscript][hbf2gf][makeindex][metafont][metapost][mft][miktex][mltex][omega]
[pdftex][psutils][scripts][source][tex][tpm][ttf2pfb][ttf2tfm][web2c] done
[5]> updmap --enable Map charter.map
[6]> pdflatex fonttest
This is pdf $\TeX$ , Version 3.141592-1.21a-2.2 (MiK $\TeX$  2.4)
entering extended mode
(fonttest.tex
LaTeX2e <2003/12/01>
... a few lines deleted
(C:\TL2005\texmf\tex\latex\psnfss\charter.sty)
No file fonttest.aux.
(C:\TL2005\texmf\tex\latex\psnfss\tl\bch.fd)
[lpsfonts.map](fonttest.aux) )8r.enc
<C:\TL2005\localtexmf\fonts\type1\public\bitstrea\charter\bchbi8a.pfb>
<C:\TL2005\localtexmf\fonts\type1\public\bitstrea\charter\bchb8a.pfb>
<C:\TL2005\localtexmf\fonts\type1\public\bitstrea\charter\bchri8a.pfb>
<C:\TL2005\localtexmf\fonts\type1\public\bitstrea\charter\bchr8a.pfb>
Output written on fonttest.pdf (1 page, 74015 bytes).
```

no: abcdefghijklmnopqrstuvwxyz (1234567890)  
 ABCDEFGHIJKLMNOPQRSTUVWXYZ  
 SC: ABCDEFGHIJKLMNOPQRSTUVWXYZ  
*it: abcdefghijklmnopqrstuvwxyz (1234567890)*  
 ABCDEFGHIJKLMNOPQRSTUVWXYZ  
**bd: abcdefghijklmnopqrstuvwxyz (1234567890)**  
 ABCDEFGHIJKLMNOPQRSTUVWXYZ  
***bi: abcdefghijklmnopqrstuvwxyz (1234567890)***  
 ABCDEFGHIJKLMNOPQRSTUVWXYZ

Figure 21.15: Bitstream Charter font sample

```
Transcript written on fonttest.log.
[7]> latex fonttest
This is e-TeX, Version 3.141592-2.2 (MiKTeX 2.4)
entering extended mode
(fonttest.tex
LaTeX2e <2003/12/01>
... a few lines deleted
(C:\TL2005\texmf\tex\latex\psnfss\charter.sty)
(fonttest.aux)
(C:\TL2005\texmf\tex\latex\psnfss\tlbch.fd)
[1] (fonttest.aux) )
Output written on fonttest.dvi (1 page, 792 bytes).
Transcript written on fonttest.log.
[8]> dvips fonttest
This is dvips(k) 5.94b Copyright 2004 Radical Eye Software (www.radicaleye.com)
' TeX output 2006.03.19:1505' -> fonttest.ps
<tex.pro><8r.enc><texps.pro>. <bchbi8a.pfb><bchb8a.pfb><bchri8a.pfb>
<bchr8a.pfb>[1]
```

After having unzipped the fonts in a subdirectory `charter` we go into that directory (step [1]). We create a directory for the PostScript Type 1 .pfb sources (step [2], note that MikTeX creates by default the `c:\localtexmf` directory to contain local additions) and copy them to that directory (step [3]). A nice feature on MikTeX is that the T<sub>E</sub>X update program `texhash` (step [4]) shows how it traverses the various directories to recreate the tree database. After adding the `charter.map` to the global font map database (step [5]) we run our small test file with `pdflatex` (step [6]). We find that the package `charter` is correctly loaded as is the corresponding font definition file `tlbch.fd`. We also see that `pdflatex` reads the `psfonts.map` file, that gives access to all font maps, including `utopia.map`, so that the program knows where to find the relevant .pfb files, which it duely includes from the directory where we deposited them earlier.

To be able to compare the installation procedure with the one we showed for Utopia on Linux we also run E<sub>T</sub><sub>X</sub> and `dvips` (steps [7] and [8]). The result of that run is shown in Figure 21.15.

It has to be noted that pdfT<sub>E</sub>X and `dvips` have their own map files. On Linux, `dvips` refers to the map file `psfonts.map` in the subdirectory `fonts/map/dvips/updmap` while pdfT<sub>E</sub>X refers to the map file `pdftex.map` in the subdirectory `fonts/map/pdftex/updmap`. On MikTeX `dvips` and pdfT<sub>E</sub>X both refer to a file with the name `psfonts.map` but the first one takes it from `dvips\config` and the latter from `pdftex\config`, both under `C:\TL2005\localtexmf`.

## 21.4 Using commercial PostScript Type 1 fonts with $\text{\LaTeX}$

Walter Schmidt maintains a set of support files (they can be downloaded from `CTAN:/fonts/psfonts/w-a-schmidt/`) for using various commercial PostScript Type 1 font families with  $\text{\LaTeX}$ . Table 21.5 gives a list (the Berry naming scheme is used). Each set consists of a zip archive and an associated text file explaining how to install the files. The distributed material does, of course, not contain the actual .pfb font instances themselves, which have to be purchased from the relevant font foundries.

<b>Adobe</b>			
Adobe Garamond	pad	Aldus	pas
Futura Light/Book/Bold/BoldCond	pfu	Stempel Garamond	peg
Frutiger	pfr	Melior	pml
Minion	pmn	Myriad	pmy
Optima	pop	Rotis	pro
Sabon	psb	Syntax	psx
<b>Berthold</b>			
ConcordeBE	poc	Berthold BaskervilleBQ	qeb
<b>Bitstream</b>			
Humanist 777 (Frutiger)	bfr	Letter Gothic	blg
Latin 725 (Meridien)	bmd	News Gothic	bng
Dutch 809 (Concorde)	boc	Zapf Humanist 601 (Optima)	bop
Zurich (Univers)	bun	Venetian 301 (Centaur)	bur
<b>Fontfont</b>			
Dingbests*	dingbests	InterOffice*	interoffice
QType Extended Book*	qtype		
<b>Linotype</b>			
Adobe Garamond LT	pad	Aldus LT	las
ITC Charter	lch	Frutiger Next	lf9
FuturaLT Light/Book/Bold/BoldCond	pfu	Stempel Garamond LT	leg
Meridien LT	lmd	Melior LT	lml
Minion	lmn	Myriad	pmy
ITC Officina Sans	lo9	Optima Nova (Regular+Italic)	ln9
Sabon LT	lsb	Syntax LT	lsx
ITC Zapf Chancery LT	lzc	Zapf Essentials LT	lze
Zapfino One + Ornaments	zap		
<b>Monotype</b>			
TimesNR, NR Seven, Small Text	mnt		
<b>Springer</b>			
SMinion + SMyriad*	sfonts		
<b>Underware</b>			
Dolly	dolly		

\* These fonts are freely available, see the accompanying .txt files.

Table 21.5: List of Walter Schmidt's PostScript Type 1 support packages

### 21.4.1 Installing Optima

As an example of how to use Walter's files, let us install his Optima bundle, which includes the  $\text{\LaTeX}$  support files required for using the three Adobe Optima PostScript Type 1 fonts *Optima*, *Optima-Medium*, and *Optima-Bold*.

**pop:** A floating field for £45. **THE SILLY FOX HAD 12345–67890 JUMPS!** —  
 ¿Are Kafka’s Schloß and Æsop’s Œuvres naïve vis-à-vis a *dæmonic phoenix’s rôle* and its *soufflés*?

**bmd:** A floating field for £45. **THE SILLY FOX HAD 12345–67890 JUMPS!**  
 — ¿Are Kafka’s Schloß and Æsop’s Œuvres naïve vis-à-vis a *dæmonic phoenix’s rôle* and its *soufflés*?

**bun:** A floating field for £45. **THE SILLY FOX HAD 12345–67890 JUMPS!** —  
 ¿Are Kafka’s Schloß and Æsop’s Œuvres naïve vis-à-vis a *dæmonic phoenix’s rôle* and its *soufflés*?

Figure 21.16: Text sample of Adobe Optima and Bitstream Meridien and Univers

### 21.4.1.1 Getting the PostScript Type 1 font instances

As mentioned, Walter’s files do not contain the font instances. In the case of Optima you would have to buy them from Adobe in PostScript Type 1 or OpenType format. In the former case you can just install the `.pfb` files, as indicated later. If you get Adobe’s OpenType variant you should extract the Type 1 information from the font, e.g., with the `cfftot1` program of the *lcdf Typetools* (see <http://www.lcdf.org/type/>). This program translates an OpenType font from its Compact Font Format (CFF) to the usual PostScript Type 1 format. It is used in Section 21.6.3 with the *Minion Pro* fonts).

### 21.4.1.2 Installing the font files

The PostScript Type 1 font files are to be named properly for use with  $\TeX$ . The list of the PostScript full name for the four font instances (with the corresponding “berry” name in parentheses) follows: *Optima* (`popr8a.pfb`), *Optima-Medium* (`popm8a.pfb`), *Optima-Bold* (`popb8a.pfb`), and *Optima-Black* (`popc8a.pfb`). These `.pfb` files should be copied to the directory tree with updates of your  $\TeX$  system (the way this is set up depends on your local setup),<sup>1</sup>

Now that we have the font glyphs, we can get the  $\TeX$  support files in the archive `pop.zip`<sup>2</sup> and install them. Proceed by placing this file at the top of the directory tree: `texmf-updates` or `local-texmf` and `unzip` it. In this way all files will end up in the appropriate directories.

In order to use the Optima fonts you have to inform the system about their presence. Therefore you should update the  $\TeX$  file database and register the `pop.map` file, which defines the association between the font metrics used by the  $\TeX$  engine and the PostScript Type 1 font glyphs to be put in the final output file. On  $\text{teTeX}$ -based systems, such as  $\TeX$  Live and  $\text{MiKTeX}$ , the following commands should be executed:

```
texhash
updmap --enable Map pop.map
```

Other  $\TeX$  distributions have similar commands for updating the system directories.

Together with the Adobe Optima font (`pop`), we also install the Bitstream Latin 725 (Meridien, Berry name: `bmd`) and the Bitstream “Zurich” (Univers) font family (Berry name: `bun`, see Table 21.5 on the preceding page) by getting the relevant files from CTAN, copying them into the update  $\TeX$  tree, and updating the databases, as shown above for Optima (first `rehash`, then `updmap` on the relevant file). A short text example for the three fonts together is displayed in Figure 21.16.

<sup>1</sup> $\TeX$  Live on Unix: `/TL2007/texmf-updates/fonts/type1/adobe/pop`  
 $\text{MiKTeX}$  on Windows `\TL2007\localtexmf\fonts\type1\adobe\pop`

<sup>2</sup>Available from CTAN: `/fonts/psfonts/w-a-schmidt/bop.zip`.



m/n	àbcdéfgHJKLMNÔP12345üyZ	m/it	àbcdéfgHJKLMNÔP12345üyZ
mc/n	àbcdéfgHJKLMNÔP12345üyZ	mc/it	àbcdéfgHJKLMNÔP12345üyZ
mx/n	àbcdéfgHJKLMNÔP12345üyZ		
mec/n	àbcdéfgHJKLMNÔP12345üyZ		
b/n	àbcdéfgHJKLMNÔP12345üyZ	b/it	àbcdéfgHJKLMNÔP12345üyZ
bc/n	àbcdéfgHJKLMNÔP12345üyZ	bc/it	àbcdéfgHJKLMNÔP12345üyZ
bx/n	àbcdéfgHJKLMNÔP12345üyZ		
bec/n	àbcdéfgHJKLMNÔP12345üyZ		
eb/n	àbcdéfgHJKLMNÔP12345üyZ	eb/it	àbcdéfgHJKLMNÔP12345üyZ
ebx/n	àbcdéfgHJKLMNÔP12345üyZ		
ub/n	àbcdéfgHJKLMNÔP12345üyZ		
ubx/n	àbcdéfgHJKLMNÔP12345üyZ		
l/n	àbcdéfgHJKLMNÔP12345üyZ	l/it	àbcdéfgHJKLMNÔP12345üyZ
lc/n	àbcdéfgHJKLMNÔP12345üyZ	lc/it	àbcdéfgHJKLMNÔP12345üyZ
lec/n	àbcdéfgHJKLMNÔP12345üyZ		

Figure 21.17: Weights and widths available for the Bitstream Univers font

We can have a closer look at the Univers font, a sans serif design of Adrian Frutiger that is by some considered as one of the greatest typographic achievements of the second half of the 20th century. The typeface comes in a large variety of weights. The weights and widths supported by the `bunivers` package are shown in Figure 21.17 (L<sup>A</sup>T<sub>E</sub>X’s NFSS scheme is used to label the samples, see also Tables 21.12 and 21.13).

## 21.5 Using TrueType fonts with pdfT<sub>E</sub>X

Some commercial T<sub>E</sub>X systems support TrueType fonts natively, for instance, Richard J. Kinch’s TrueT<sub>E</sub>X (<http://www.trueteX.com>) on Microsoft Windows. XeT<sub>E</sub>X (<http://scripts.sil.org/xetex>), being developed on Apple Macintosh OS X, but with a port to Linux being announced (Summer 2006), has built-in support for OpenType fonts, and thus can also handle TrueType fonts. On the other hand, some work has been done to translate TrueType fonts to PostScript Type 1 and install the latter, e.g., Harald Harders’s *Using TrueType fonts with T<sub>E</sub>X via PostScript Type 1 format* (<http://www.tug.org/tex-archive/info/TrueType/>).

### 21.5.1 A predefined setup for using Microsoft Windows TrueType fonts

Christophe Caignaert developed a procedure to generate the L<sup>A</sup>T<sub>E</sub>X NFSS files needed for using TrueType files with pdf<sub>l</sub>at<sub>e</sub>x. He prepared T<sub>E</sub>X font metrics (`.tfm`), font definition (`.fd`) and map (`.map`) files for many Microsoft Windows fonts.<sup>1</sup>

<sup>1</sup>They can be downloaded from <http://c.caignaert.free.fr/WindowsFonts.zip>. The scripts used for generating `.fd` and `.map` files are available as <http://c.caignaert.free.fr/shells.zip>. The procedure for installing a TrueType font with L<sup>A</sup>T<sub>E</sub>X or pdf<sub>l</sub>at<sub>e</sub>x, including kerning and ligature tables, is explained in <http://c.caignaert.free.fr/Install-ttf-Font.pdf>. Christophe also developed some material for the *Dafont* set of freeware and shareware fonts ([www.dafont.com](http://www.dafont.com)), with installation instructions (<http://c.caignaert.free.fr/HowTo.pdf>) and the relevant `.tfm`, `.map`, and `.fd` files (<http://c.caignaert.free.fr/HowTo-Dafont.pdf>).



In Figure 21.18 on the following page we typeset a variant of the text of Figure 21.2 on page 37 using pdfLatex and TrueType fonts that are available on most Microsoft Windows systems. We downloaded Christophe Caignaert's file `WindowsFonts.zip` (see above) and placed the font and map files in the relevant directories, updated the T<sub>E</sub>X files database and ran the following file (`ttftest.tex`) with pdfLatex (the font labels are explained in the figure.)

```
\newcommand{\MyText}[1]{\fontfamily{#1}\selectfont\fbx{#1:%
} A floating field for \pounds45. \textbf{!`THE SILLY FOX HAD
12345--67890 JUMPS!} --- ?`Are Kafka's Schlo{\ss} and {\AE}sop's
{\OE}uvres na{"i}ve vis-\'{a}-vis a \emph{d{\ae}monic ph{\oe}nix's
\textbf{r\^{o}le} and its souffl\'{e}s?}\par}
\MyText{arl}\MyText{bka}\MyText{bko}\MyText{crr}\MyText{ctg}\MyText{frk}
\MyText{grg}\MyText{lucc}\MyText{mgm}\MyText{ppt}\MyText{tim}\MyText{vdn}
```

Running pdfLatex on this file displays the following log with MiK<sub>T</sub>E<sub>X</sub> on Microsoft Windows.

```
This is pdfTeX, Version 3.141592-1.21a-2.2
(MiKTeX 2.4) entering extended mode
(ttftest.tex LaTeX2e <2003/12/01>
Babel <v3.8g> and hyphenation patterns for english, dumylang,
nohyphenation, german, ngerman, french, loaded.
(C:\TL2005\texmf\tex\latex\base\article.cls
Document Class: article 2004/02/16 v1.4f
Standard LaTeX document class
(C:\TL2005\texmf\tex\latex\base\size10.clo)
(C:\TL2005\texmf\tex\latex\base\fontenc.sty
(C:\TL2005\texmf\tex\latex\base\tlenc.def))
No file ttftest.aux.
(C:\TL2005\localtexmf\tex\latex\winfonts\tlarl.fd)
(C:\TL2005\localtexmf\tex\latex\winfonts\tlbka.fd)
(C:\TL2005\localtexmf\tex\latex\winfonts\tlbko.fd)
(C:\TL2005\localtexmf\tex\latex\winfonts\tlcrr.fd)
(C:\TL2005\localtexmf\tex\latex\winfonts\tlctg.fd)
(C:\TL2005\localtexmf\tex\latex\winfonts\tlfrk.fd)
(C:\TL2005\localtexmf\tex\latex\winfonts\tlgrg.fd)
(C:\TL2005\localtexmf\tex\latex\winfonts\tllucc.fd)
(C:\TL2005\localtexmf\tex\latex\winfonts\tlmgm.fd)
(C:\TL2005\localtexmf\tex\latex\winfonts\tlppt.fd)
(C:\TL2005\localtexmf\tex\latex\winfonts\tltim.fd)
(C:\TL2005\localtexmf\tex\latex\winfonts\tlvdn.fd)
[1{psfonts.map}] (ttftest.aux) {T1-WGL4.enc}
<verdanaz.ttf><verdanai.ttf><verdanab.ttf><verdana.ttf>
<timesbi.ttf><timesi.ttf><timesbd.ttf><times.ttf>
<perpetuabi.ttf><perpetuait.ttf><perpetuabd.ttf><perpetua.ttf>
<garabd.ttf><garait.ttf><gara.ttf>
<ltypeo.ttf><ltypebo.ttf><ltypeb.ttf><ltype.ttf>
<georgiaz.ttf><georgiai.ttf><georgiab.ttf><georgia.ttf>
<fradmit.ttf><frabkit.ttf><fradm.ttf><frabk.ttf>
<gothicbi.ttf><gothici.ttf><gothicb.ttf><gothic.ttf>
<courbi.ttf><couri.ttf><courbd.ttf><cour.ttf>
<bookosbi.ttf><bookosi.ttf><bookosb.ttf><bookos.ttf>
<antquabi.ttf><antquai.ttf><antquab.ttf><bkant.ttf>
<arialbi.ttf><ariali.ttf><arialbd.ttf><arial.ttf>
Output written on ttftest.pdf (1 page, 578030 bytes).
```

**arl:** A floating field for £45. **¡THE SILLY FOX HAD 12345–67890 JUMPS!** — ¿Are Kafka's Schloß and Æsop's Œuvres naïve vis-à-vis a *dæmonic phœnix's rôle and its soufflés?*

**bka:** A floating field for £45. **¡THE SILLY FOX HAD 12345–67890 JUMPS!** — ¿Are Kafka's Schloß and Æsop's Œuvres naïve vis-à-vis a *dæmonic phœnix's rôle and its soufflés?*

**bko:** A floating field for £45. **¡THE SILLY FOX HAD 12345–67890 JUMPS!** — ¿Are Kafka's Schloß and Æsop's Œuvres naïve vis-à-vis a *dæmonic phœnix's rôle and its soufflés?*

**crr:** A floating field for £45. **! 'THE SILLY FOX HAD 12345--67890 JUMPS!** --- ? 'Are Kafka's Schloß and Æsop's Œuvres naïve vis-à-vis a *dæmonic phœnix's rôle and its soufflés?*

**ctg:** A floating field for £45. **¡THE SILLY FOX HAD 12345–67890 JUMPS!** — ¿Are Kafka's Schloß and Æsop's Œuvres naïve vis-à-vis a *dæmonic phœnix's rôle and its soufflés?*

**frk:** A floating field for £45. **¡THE SILLY FOX HAD 12345–67890 JUMPS!** — ¿Are Kafka's Schloß and Æsop's Œuvres naïve vis-à-vis a *dæmonic phœnix's rôle and its soufflés?*

**grg:** A floating field for £45. **¡THE SILLY FOX HAD 12345–67890 JUMPS!** — ¿Are Kafka's Schloß and Æsop's Œuvres naïve vis-à-vis a *dæmonic phœnix's rôle and its soufflés?*

**lucc:** A floating field for £45. **¡THE SILLY FOX HAD 12345–67890 JUMPS!** — ¿Are Kafka's Schloß and Æsop's Œuvres naïve vis-à-vis a *dæmonic phœnix's rôle and its soufflés?*

**mgm:** A floating field for £45. **¡THE SILLY FOX HAD 12345–67890 JUMPS!** — ¿Are Kafka's Schloß and Æsop's Œuvres naïve vis-à-vis a *dæmonic phœnix's rôle and its soufflés?*

**ppt:** A floating field for £45. **¡THE SILLY FOX HAD 12345–67890 JUMPS!** — ¿Are Kafka's Schloß and Æsop's Œuvres naïve vis-à-vis a *dæmonic phœnix's rôle and its soufflés?*

**tim:** A floating field for £45. **¡THE SILLY FOX HAD 12345–67890 JUMPS!** — ¿Are Kafka's Schloß and Æsop's Œuvres naïve vis-à-vis a *dæmonic phœnix's rôle and its soufflés?*

**vdn:** A floating field for £45. **¡THE SILLY FOX HAD 12345–67890 JUMPS!** — ¿Are Kafka's Schloß and Æsop's Œuvres naïve vis-à-vis a *dæmonic phœnix's rôle and its soufflés?*

arl	Arial	arial.ttf	bka	Book Antiqua	bkant.ttf
bko	BookMan Old Style	bookos.ttf	crr	Courier New	cour.ttf
ctg	Century Gothic	gothic.ttf	frk	Franklin Gothic Book	frabk.ttf
grg	Georgia	georgia.ttf	lucc	Lucida Sans Typewriter Regular	ltype.ttf
mgm	Garamond	gara.ttf	ppt	Perpetua	perpetua.ttf
tim	Times New Roman	times.ttf	vdn	Verdana	verdana.ttf

Figure 21.18: Example of using TrueType fonts with pdfflatex

Transcript written on ttftest.log.

The T1 encoding is used and the various font definition files are loaded in the same sequence as the fonts are referenced in the input file. After the typesetting stage pdfLatex has to get hold of the font glyphs, and hence consults the `psfonts.map` file, where it finds the actual TrueType files names and the encoding map (T1-WGL4.enc) to be used for outputting the correct characters. Finally, all TrueType font files are loaded to extract the actual glyphs and transfer them to the output PDF file `ttftest.pdf`.

## 21.5.2 Doing it yourself: under the hood

For those interested in setting up a TrueType font whose  $\TeX$  support files are not in Christophe Caignaert's distribution, as described in the previous section, we describe now briefly how to proceed.<sup>1</sup>

Let us suppose we want to use the *Times New Roman* TrueType font file `times.ttf` (this file is present in the system font directory of Microsoft Windows systems and can be copied to a local directory for convenience). First we make the  $\TeX$  font metric file with the program `ttf2tfm`. This program uses the encoding T1-WGL4.enc (-T switch) and generates the *virtual properties list* file `timmn8t.vpl` (-v switch). The `.tfm` file `times.tfm` has no kerning or ligature information.

```
> ttf2tfm times.ttf -q -T T1-WGL4.enc -v timnr8t.vpl times.tfm
ttf2tfm: WARNING: Cannot find character 'compwordmark'
          specified in input encoding.
ttf2tfm: WARNING: Cannot find character 'perthousandzero'
          specified in input encoding.
ttf2tfm: WARNING: Cannot find character 'dotlessj'
          specified in input encoding.
ttf2tfm: WARNING: Cannot find character 'ff'
          specified in input encoding.
ttf2tfm: WARNING: Cannot find character 'ffi'
          specified in input encoding.
ttf2tfm: WARNING: Cannot find character 'ffl'
          specified in input encoding.
ttf2tfm: WARNING: Cannot find character 'visualspace'
          specified in input encoding.
times    times.ttf Encoding=T1-WGL4.enc
```

Ligatures and kerning is added by running the `.vpl` file through the `vptovf` program, which generates the desired `.tfm` file `timmn8t.tfm`.

```
> vptovf timmn8t.vpl timmn8t.vf timmn8t.tfm
This is VPTovF, Version 1.5 (MiKTeX 2.4)
'000 '001 '002 '003 '004 '005 '006 '007
... many lines deleted
'367 '370 '371 '372 '373 '374 '375 '376
'377.
I had to round some heights by 0000010 units.
I had to round some depths by 0000001 units.
```

This procedure for generating the `.tfm` file has to be repeated for all variants of a given TrueType font, each time using the correct “Berry” name for the `.tfm` filename (see Section 21.7). These names

<sup>1</sup>See also Damir Rakityansky's web page *Using TrueType fonts with  $\TeX$  ( $\mathcal{E}\TeX$ ) and pdf $\TeX$  (pdf $\mathcal{E}\TeX$ )* (<http://www.radamir.com/tex/ttf-tex.htm>).

for *Times New Roman* can be seen as the first entry on the four lines that have been entered in the file `psfonts.map`. The third entry on each line is the actual name of the TrueType font on the system.

```
timmn8t Timons <times.ttf <T1-WGL4.enc
timbn8t TimonsBold <timesbd.ttf <T1-WGL4.enc
timmit8t TimonsItalique <timesi.ttf <T1-WGL4.enc
timbit8t TimonsBoldItalique <timesbi.ttf <T1-WGL4.enc
```

These “Berry” names are also used in the font definition file which defines for each font family the mapping between fonts instances and  $\text{\TeX}$  font parameters (shapes and series). This file can be compared to the one for the PostScript Type 1 font *Times Roman* (see Section 21.2.7), where we also generated small capitals and slanted variants with the PostScript machinery, whereas here we merely substitute these variants with bold upright (lines 10 and 12) and italic (lines 11 and 13). The bold extended series is substituted with the bold variant (lines 15–18).

```
1 \ProvidesFile{t1tim.fd}[2003/11/11 font definitions for T1/tim.]
2
3 \DeclareFontFamily{T1}{tim}{}
4
5 \DeclareFontShape{T1}{tim}{m}{n}{<-> timmn8t}{}
6 \DeclareFontShape{T1}{tim}{b}{n}{<-> timbn8t}{}
7 \DeclareFontShape{T1}{tim}{m}{it}{<-> timmit8t}{}
8 \DeclareFontShape{T1}{tim}{b}{it}{<-> timbit8t}{}
9
10 \DeclareFontShape{T1}{tim}{m}{sc}{<->ssub * tim/b/n}{}
11 \DeclareFontShape{T1}{tim}{m}{sl}{<->ssub * tim/m/it}{}
12 \DeclareFontShape{T1}{tim}{b}{sc}{<->ssub * tim/b/n}{}
13 \DeclareFontShape{T1}{tim}{b}{sl}{<->ssub * tim/b/it}{}
14
15 \DeclareFontShape{T1}{tim}{bx}{n}{<->ssub * tim/b/n}{}
16 \DeclareFontShape{T1}{tim}{bx}{sc}{<->ssub * tim/b/n}{}
17 \DeclareFontShape{T1}{tim}{bx}{sl}{<->ssub * tim/b/it}{}
18 \DeclareFontShape{T1}{tim}{bx}{it}{<->ssub * tim/b/it}{}

```

As explained previously for the case of installing PostScript Type 1 files these  $\text{\TeX}$  support files must be copied below the `localtexmf` directory to the relevant places<sup>1</sup>. Once the files are in their correct location and we updated the  $\text{\TeX}$  file database (`rehash`) (the have entered the map entries for the Truetype files directly into `psfonts.map` so no action is required here) we can use the setup on a small test file `fonttest.tex`. The typeset result is displayed in Figure 21.19 on the facing page.

```
> pdflatex fonttest.tex
This is pdfTeX, Version 3.141592-1.21a-2.2 (MiKTeX 2.4)
entering extended mode
(fonttest.tex
LaTeX2e <2003/12/01>
Babel <v3.8g> and hyphenation patterns for english, dumylang, nohyphenation, ge
rman, ngerman, french, loaded.
(C:\TL2005\texmf\tex\latex\base\article.cls
Document Class: article 2004/02/16 v1.4f Standard LaTeX document class
(C:\TL2005\texmf\tex\latex\base\size10.clo))
```

<sup>1</sup>For instance, for the `.fd` file, `localtexmf\tex\latex\winfonts\t1tim.fd`, for the Truetype files `localtexmf\fonts\truetype\winfonts\timesnewroman\times.ttf`, etc., for the `.tfm` files (and similarly for the `.vf` files) `localtexmf\fonts\tfm\winfonts\timesnewroman\timmn8t.tfm`, etc.

no: abcdefghijklmnopqrstuvwxyz (1234567890)  
 ABCDEFGHIJKLMNOPQRSTUVWXYZ  
*it: abcdefghijklmnopqrstuvwxyz (1234567890)*  
*ABCDEFGHIJKLMNOPQRSTUVWXYZ*  
**bd: abcdefghijklmnopqrstuvwxyz (1234567890)**  
**ABCDEFGHIJKLMNOPQRSTUVWXYZ**  
*bi: abcdefghijklmnopqrstuvwxyz (1234567890)*  
*ABCDEFGHIJKLMNOPQRSTUVWXYZ*

Figure 21.19: TrueType font sample of Microsoft Times New Roman typeset with pdfTeX

```
(C:\TL2005\texmf\tex\latex\base\fontenc.sty
(C:\TL2005\texmf\tex\latex\base\tlenc.def)) (fonttest.aux)
(C:\TL2005\localtexmf\tex\latex\winfonts\tl\tim.fd)
[lpsfonts.map] (fonttest.aux) )T1-WGL4.enc
<timesbi.ttf><timesbd.ttf><timesi.ttf><times.ttf>
Output written on fonttest.pdf (1 page, 133017 bytes).
Transcript written on fonttest.log.
```

### 21.5.3 Unicode support with *Cyberbit*

Unicode contains code points for most characters used in current (and past) languages. Until recently fonts were developed with the characters needed to support a given family of languages that use the same character set. Hence there exist many different type faces for languages written with Latin, Cyrillic, Greek, Indian, Japanese, Chinese, Korean, etc. characters. Nowadays, with support for Unicode becoming more generally available, font sets including many hundreds of characters in several different alphabets are appearing, especially in the OpenType format.

Bitstream's *Cyberbit* TrueType font, which has been available for several years, is an international font, containing characters from many languages, with each character encoded at its Unicode value. *Cyberbit*, developed originally by Bitstream to provide Unicode Consortium members with a test font, can be distributed freely for testing and other non-commercial purposes. We shall use the *Cyberbit* font with pdf<sub>l</sub>at<sub>e</sub>x as an example of typesetting a multi-lingual document.

#### 21.5.3.1 Getting and installing the *Cyberbit* font

We first download the *Cyberbit* font<sup>1</sup> and install it on the system.

```
[1]> unzip cyberbit.zip
[2]> rename Cyberbit.ttf cyberbit.ttf
[3]> copy cyberbit.ttf \tl2005\localtexmf\fonts\truetype\bitstream
```

Next we generate .t<sub>f</sub>m T<sub>E</sub>X metric and .enc encoding files, and copy them to the right place, as follows.

```
[1]> ttf2tfm cyberbit.ttf -w cyberbit@Unicode.sfd@ > cyberbit.log
[2]> copy *.tfm \tl2005\localtexmf\fonts\tfm\bitstream\cyberbit
[3]> copy *.enc \tl2005\localtexmf\fonts\enc\cyberbit
```

<sup>1</sup>See <ftp://ftp.netscape.com/pub/communicator/extras/fonts/windows/cyberbit.zip>.

This should copy a total of 165 .tfm and .enc files to their respective directories. Next we create the file `\tl2005\localtexmf\ttf2tfm\base\ttfonds.map` and we enter the following line in it:

```
cyberbit@Unicode@ cyberbit.ttf
```

Now we create the file `\tl2005\localtexmf\web2c\updmap.cfg` and put the following line in it:

```
Map cyberbit.map #localtexmf/fonts/map/cyberbit.map
```

We download the Cyberbit map ([delloye.free.fr/cyberbit.map](http://delloye.free.fr/cyberbit.map)) and font definition ([delloye.free.fr/c70cyberbit.fd](http://delloye.free.fr/c70cyberbit.fd)) files and copy them to the right place.

```
[1]> copy cyberbit.map \tl2005\localtexmf\fonts\map\pdfTeX
[2]> copy c70cyberbit.fd \tl2005\localtexmf\tex\latex\cyberbit
```

Finally, we run `updmap` and refresh the  $\TeX$  database (`texhash`).

### 21.5.3.2 Typesetting a multi-lingual document

After installing the Cyberbit font, we are ready to test it with a  $\TeX$  file that contains multiple languages. We take (minimal) advantage of Werner Lemberg's CJK package to indicate we are using the `utf8` encoding and the Cyberbit font. We used the first article of the *Universal Declaration of Human Rights*<sup>1</sup> in seven languages as a test. The  $\TeX$  input file follows.

```
\documentclass[12pt,a4paper]{article}
\parindent0pt
\parskip\baselineskip
\newenvironment{Arabic}
{\TeXeTstate=1\relax
 \let\myeverypar\everypar
 \newtoks\everypar
 \everypar\expandafter{\the\myeverypar}
 \myeverypar{\the\everypar\beginR}\beginR}
{\TeXeTstate=0\relax}
\pagestyle{empty}
\usepackage{CJK}
\begin{document}
\begin{CJK}{UTF8}{cyberbit}
\begin{description}
\item[English] \input udhreng
\item[French] \input udhrfrn
\item[Russian] \input udhrrus
\begin{Arabic}
\item[Arabic] \input udhrarz
\end{Arabic}
\item[Chinese] \input udhrchn
\item[Japanese] \input udhrjpn
\item[Korean] \input udhrkkn
\end{description}
\end{CJK}
\end{document}
```

<sup>1</sup>Available at <http://www.unicode.org/udhr/> in over 300 languages.

As we want to typeset Arabic from right to left, we define an `Arabic` environment, which initiates the relevant e- $\text{\TeX}$  extension, then ensures that every paragraph starts right-to-left typesetting. The various files that are input contain for each language two paragraphs: the title and the first article of the *Declaration*. When we run this file (`udhrall.tex`) with `pdflatex` we get the result shown in Figure 21.20 on the next page together with following log output.

```
This is pdfTeX, Version 3.141592-1.21a-2.2 (MiKTeX 2.4)
entering extended mode
(udhrall
LaTeX2e <2003/12/01>
Babel <v3.8g> and hyphenation patterns for english, dumylang,
  nohyphenation, german, ngerman, french, loaded.
(C:\t12005\texmf\tex\latex\base\article.cls
Document Class: article 2004/02/16 v1.4f Standard LaTeX document class
(C:\t12005\texmf\tex\latex\base\size12.clo))
(C:\t12005\texmf\tex\latex\CJK\CJK.sty
(C:\t12005\texmf\tex\latex\CJK\mule\MULEenc.sty)
(C:\t12005\texmf\tex\latex\CJK\CJK.enc))
No file udhrall.aux.
(C:\t12005\texmf\tex\latex\CJK\UTF8\UTF8.bdg)
(C:\t12005\texmf\tex\latex\CJK\UTF8\UTF8.enc)
(C:\t12005\texmf\tex\latex\CJK\UTF8\UTF8.chr)
(udhreng.tex) (udhrfrn.tex
(C:\t12005\localtexmf\tex\latex\cyberbit\c70cyberbit.fd))
(udhrrus.tex) (udhrrarz.tex) (udhrchn.tex) (udhrrjpn.tex)
(udhrkkn.tex) [1{psfonts.map}] (udhrall.aux) )
(see the transcript file for additional information)
{cyberbitd6.enc}<cyberbit.ttf>{cyberbitbc.enc}<cyberbit.ttf>
{cyberbitc2.enc}<cyberbit.ttf>{cyberbitcc.enc}<cyberbit.ttf>
[ ... ]
<C:\t12005\texmf\fonts\type1\bluesky\cm\cmr12.pfb>
<C:\t12005\texmf\fonts\type1\bluesky\cm\cmbx12.pfb>
Output written on udhrall.pdf (1 page, 374564 bytes).
Transcript written on udhrall.log.
```

After loading the CJK package files the Cyberbit font definitions, and our seven utf8 encoded source files are input. Then many lines follow indicating that the encoding file for a given character range (`cyberbitij`) is read and the needed characters are extracted from `cyberbit.ttf`.

Of course, for more advanced typesetting (contextual analysis to guarantee correct Arabic ligatures, etc.), one should use *Omega* (<http://omega.enstb.org/>) or a similar system, that fully supports OpenType fonts to handle the needed presentational forms.

## 21.6 Installing OpenType Fonts in $\text{\LaTeX}$

$\text{\LaTeX}$  can extract metric information (including kerning and ligatures) directly from OpenType fonts (using e.g., OpenType's `GPOS` and `GSUB` tables, which manage, respectively, character positioning, and glyph substitution). However, when you want to use the basic  $\text{\TeX}$  engine, the you have to live with font support files (`.tfm` and `.vf`) which are limited to 256 character positions. Hence to access the several hundreds or even thousands of characters in a large OpenType font we must generate many `.tfm` and `.vf` files, each containing the metric information about a small part of the OpenType font. The program `oftotfm` can perform this task (see below).

**English** Universal Declaration of Human Rights

All human beings are born free and equal in dignity and rights. They are endowed with reason and conscience and should act towards one another in a spirit of brotherhood.

**French** Déclaration universelle des droits de l'homme

Tous les êtres humains naissent libres et égaux en dignité et en droits. Ils sont doués de raison et de conscience et doivent agir les uns envers les autres dans un esprit de fraternité.

**Russian** Всеобщая декларация прав человека

Все люди рождаются свободными и равными в своем достоинстве и правах. Они наделены разумом и совестью и должны поступать в отношении друг друга в духе братства.

**Arabic**

الإعلان العالمي لحقوق الإنسان

يولد جميع الناس أحراراً متساوين في الكرامة والحقوق. وقد وهبوا عقلاً وضميراً وعلمهم أن يعامل بعضهم بعضاً بروح الإخاء.

**Chinese** 世界人权宣言

人人生而自由，在尊严和权利上一律平等。他们赋有理性与良心，并应以兄弟关系的精神相对待。

**Japanese** 『世界人権宣言』

すべての人間は、生まれながらにして自由であり、かつ、尊厳と権利とについて平等である。人間は、理性と良心とを授けられており、互いに同胞の精神をもって行動しなければならない。

**Korean** 세계인권선언

모든 인간은 태어날 때부터 자유로우며 그 존엄과 권리에 있어 동등하다. 인간은 천부적으로 이성과 양심을 부여받았으며 서로 형제애의 정신으로 행동하여야 한다.

Figure 21.20: Multi-lingual Unicode document typeset with the Cyberbit TrueType font

An interesting general introduction to the topic of exploiting the typographic riches of large fonts is Sivan Toledo's article in TugBoat [19].

In the United States Adobe sells a low-cost CD-ROM *Adobe Type Classics for Learning* (see <http://www.adobe.com/education/products/typeclassics.html>), which contains a mix of 400 Western fonts and 26 Japanese fonts in OpenType format, including “classics,” such as Adobe *Garamond*, *Minion*, *Myriad*, and *Caslon*. Because of the availability of this CD-ROM, most work related to making OpenType fonts work with  $\text{\LaTeX}$  use fonts available on that CD-ROM.



### 21.6.1 Creating the $\TeX$ font instances

Michael Saunders worked out a procedure to install and use OpenType fonts with  $\TeX$ .<sup>1</sup> He generated an NFSS package to access OpenType features. He basically separated series into weight and width axes and shape into two independent axes and he introduced several macros for setting fractions. We shall describe this work in somewhat more detail.

Michael developed his tools for the Adobe *Caslon Pro* font (from the *Adobe Type Classics for Learning* CD-ROM), which comes in three weights divided into six files:

ACaslonPro-Bold.otf	ACaslonPro-BoldItalic.otf
ACaslonPro-Semibold.otf	ACaslonPro-SemiboldItalic.otf
ACaslonPro-Regular.otf	ACaslonPro-Italic.otf

Eddie Kohler's `otftotfm` tool (<http://www.lcdf.org/type/otftotfm.1.html>) was used to prepare the installation. Previously, with some operating systems limiting filenames to eight characters, it was common practice to use Berry naming scheme (see Section 21.7). Nowadays this limit no longer applies, so that it seems appropriate to just use the fontnames given by the font vendors. For the encoding `LY1` (see Section 7.11.4, and in particular Table 7.33 of [15]) has been used since it seems to be best suited for OpenType.

OpenType and  $\TeX$ 's NFSS support many features. For the case of *Caslon Pro* Micheal has classified them as two axes plus ornaments. (The feature that must be used to activate the given font behavior is given between square brackets; for details see the description of `otftotfm` at the URL given previously. The list of available features for a font can be obtained by the program `otfinfo`, see Section 21.6.3)

#### Major axis

- Roman (`rm`, `r`): normal, upright text.
- Italic (`it`, `i`): slanted, stylized text used for emphasis.
- Slanted (`sl`, `s`): Roman letter forms slanted to match the Italics [`-S slt`, `slt` being the tangent of slant of the Italics]
- UprightItalic (`ui`, `u`): Italic letter forms slanted backwards to match the Romans. Mainly useful in displays. [`-S -amt`]

#### Minor axis

- regular (`rg`): usual, normal shape [`-fpnum -fonum`];
- all-caps (`ac`): alters the spacing around caps and the punctuation to harmonize with capitals [`-fcpsp -fcase -fpnum -flnum`];
- small-caps (`sc`): specially designed capital letter forms that harmonize well with lowercase [`-fsmcp`];
- swashes (`sw`): calligraphic forms used mostly in displays [`-fswsh`];
- inferior figures (`in`): small, low numbers [`-fsinf`];
- superior figures (`su`): superscripts, e.g., for footnote marks and ordinal numbers [`-fsups -fordn`];
- numerators (`nu`): for use in fractions [`-fnumr`];
- denominators (`de`): for use in fractions [`-fdnom`];
- proportional old-style figures (`po`): variable-width Arabic lowercase numerals with different heights [`-fpnum -fonum`];
- proportional lining (`pl`): variable-width Arabic numerals with identical heights that work well with uppercase [`-fpnum -flnum`];

<sup>1</sup>His webpage <http://members.fortunecity.com/odradek5/otf-LaTeX/index.html> documents his finding.

- tabular old-style figures (`to`): identical-width Arabic lowercase numerals with different heights [`-tnumf -fonum`];
- tabular lining figures (`tl`): identical-width Arabic uppercase numerals with identical heights [`-tnumf -flnum`].

**Ornaments** [`-formm`]

By using an appropriate switch one can select the default style of digits to use with the shapes `rg`, `ac`, `sc`, and `sw`. Usually one has proportional lining figures with all-caps and swashes (often used in displays) and proportional old-styles in normal text and with small-caps. Tabulars are only for special situations where alignments are important, such as tables and lists (see also Figure 21.21 on page 64 for examples with *Minion Pro*).

One of the `otftotfm` commands (split artificially over two lines for lack of space) that was used to set up the files for *Caslon Pro* follows.<sup>1</sup>

```
otftotfm -a -e texnansx ACaslonPro-Regular.otf -fcpsp -fcase -fpnum -flnum \
-fkern -fliga --altselector-char=@ --force LY1--ACaslonPro-RegularAC
```

This will create the `.tfm` file `LY1-ACaslonPro-RegularAC.tfm`, the `.vf` file `LY1-ACaslonPro-RegularAC.vf`, plus an encoding file for dvips. Standard switches were used to enable ligatures and kerning. The `-a` switch lets `otftotfm` place everything where it belongs in the  $\TeX$  directory tree. The `-altselector-char` switch specifies the character (\*) that gives access to alternate stylistic forms of the same character if they exist, e.g., a calligraphic variant or an historical ligature. The `texnansx` encoding (`-e` switch) corresponds to  $\mathbb{E}\TeX$ 's `LY1`, a variant of `T1` (it is documented at `CTAN:info/fontname/html/texnansx.html`.)

### 21.6.1.1 Preparing the font description (.fd) and package files

The font description file `LY1ACaslonPro.fd` starts with<sup>2</sup>

```
\ProvidesFile{LY1ACaslonPro.fd}[2005/07/13 Adobe Caslon Pro font definitions]
\DeclareFontFamily{LY1}{ACaslonPro}{}
\DeclareFontShape{LY1}{ACaslonPro}{mdnw}{rrg}{ <-> LY1--ACaslonPro-Regular }{}
\DeclareFontShape{LY1}{ACaslonPro}{mdnw}{rac}{ <-> LY1--ACaslonPro-RegularAC }{}
\DeclareFontShape{LY1}{ACaslonPro}{mdnw}{rsc}{ <-> LY1--ACaslonPro-RegularSC }{}
```

We notice the regular (`rrg`), all-caps (`rac`), and small-caps (`rsc`) definitions, which refer to the long filenames shown. This `.fd` file is placed in the  $\TeX$  tree structure (e.g., in our case `/TL2005/texmf-var/tex/latex/lcdftools`). This file defines low-level commands for font series, shapes, etc. For end users a high-level interface defined in an ad-hoc package file can contain commands like `\iscshape` (and its equivalent with an argument `\textisc`) for Italic small-caps, `\ltweight` (`\textlt`) for light weight, ornaments can be accessed with an `\ornament` command, etc.. Other nice features can include footnotemarks using superior digits, simple and compound vulgar fractions, and time signatures, etc..

Of course, with Greek, Cyrillic (and Arabic and Hebrew) characters further encoding and typesetting issues have to be addressed. Moreover, when optical variants (i.e., font source files at two or more different sizes) are present, the commands in the `.fd` file must be modified to use the appropriate size file at a given character height.

<sup>1</sup>The file at <http://members.fortunecity.com/odradek5/otf-LaTeX/otftotfm.commands.txt> contains all the `otftotfm` commands needed to install the three weights with the 49 options.

<sup>2</sup>The full file is at <http://members.fortunecity.com/odradek5/otf-LaTeX/LY1ACaslonPro.fd.txt>.

## 21.6.2 Using the LCDF Typetools

John Owens<sup>1</sup> has written a procedure to install Adobe OpenType fonts for use with  $\TeX$ . It also uses the `otftotfm` program which it calls inside a wrapper script (written in Python) to simplify the installation method and allow for easy extension to other fonts and setups.

For a given font family John's installation script<sup>1</sup> automates the loop over the many calls to `otftotfm` (which only takes a single set of options and one font file at a time) for each font instance of the various series, shapes, and options (regular, italic, bold, small caps, old-style figures, etc.) of the OpenType font.

The procedure was tested on Adobe Caslon without opticals and on Adobe Minion with opticals.

Each font places its options in the `typefaces` hash, whose fields are `typefaceoptions` (applied to all font files, such as kerning and ligatures), `options` (a list of options in turn to be applied to each file), `prefix` (the first three characters of the Berry naming scheme), optical sizes (when applicable), and vendor and typeface strings. For instance, here is the entry for *Minion Pro*:

```
typefaces = { 'Minion Pro':
  { 'typefaceoptions': ['kern', 'liga'],
    'options': ['smcp', 'onum'],
    'prefix': 'pmn',
    'vendor': 'adobe',
    'typeface': 'Minion',
    'opticals': { 'caption': '-8.4',
                  'regular': '8.5-13.0',
                  'subhead': '13.1-19.9',
                  'display': '20-',
                  }
  },
}
```

The script looks at the font's filename to decide the characteristics of the font (italic, bold, semibold, condensed, etc.). It can generate either the Berry filename based on these characteristics or long file-names, where the latter are preferred as more readable. The `LY1` (8 $\gamma$ ) encoding is used.

The argument to the script is a single path. If it corresponds to a directory, all files in that directory will be used, otherwise it will be globbed to generate one or more directories, and all files in each of those directories will be used. The script also calls `updmap` and `texhash` upon completion. Finally, the `.fd` and `.sty` are generated.

### 21.6.2.1 Font definition (`.fd`) files

Each font family must have an associated font definition file which maps font properties to fontnames. As the `LY1` encoding was used file names are of the type `ly1BBBx.fd` and `ly1PPPj.fd`, where `BBB` is the Berry prefix, "x" is for expert fonts but no old-style figures, and "j" indicates old-style figures. Internally the `.fd` files use long (vendor-supplied) file names, as the following excerpt of `ly1pacx.fd` shows.

```
% Autogenerated by ./otfinst.py on 2005/06/28
\ProvidesFile{ly1pacx.fd}[2005/06/28 LY1/Adobe Caslon Pro]

\DeclareFontFamily{LY1}{pacx}{}
\DeclareFontShape{LY1}{pacx}{sb}{sc}{
  <-> LY1-ACaslonPro-Semibold-kern-liga-smcp}{
\DeclareFontShape{LY1}{pacx}{sb}{sl}{ <-> sub * pacx/sb/it}{}
```

<sup>1</sup>Available at <http://www.ece.ucdavis.edu/~jowens/code/otfinst/otfinst.py> or in the directory CTAN: fonts/utilities/otfinst.

```

\DeclareFontShape{LY1}{pacx}{sb}{it}{
  <-> LY1-ACaslonPro-SemiboldItalic-kern-liga}{}
\DeclareFontShape{LY1}{pacx}{sb}{n}{
  <-> LY1-ACaslonPro-Semibold-kern-liga}{}

```

### 21.6.2.2 Package files (**sty**)

The package file declares the font family that  $\text{\LaTeX}$  should use in the document. There are two options, `oldstyle` (default) and `lining`, for using old-style or lining numerals, respectively.

John Owens makes the following example files available:

- Adobe Minion Pro: `lylpmnx.fd`, `lylpmnj.fd`, and `minion.sty`. In this case several “opticals” (multiple font files for each font shape) are available so that in the `.fd` files the appropriate font file is chosen depending on the size of the output.
- Adobe Caslon Pro: `lylpacx.fd`, `lylpacj.fd`, and `acaslon.sty`.
- Adobe Garamond Pro: `lylpadx.fd`, `lylpadj.fd`, and `agaramond.sty`.

In general OpenType fonts (and `otftotfm`) support many more options (e.g., variable width digits, fractions), but these are at present not supported by John Owens’s script. This is mostly due to the fact that such features are at present only poorly or not at all supported by  $\text{\LaTeX}$ .

### 21.6.3 Using the Minion Pro OpenType font

Minion Pro is an Adobe Original typeface designed by Robert Slimbach. This typeface is inspired by classical, old style typefaces of the late Renaissance, a period of elegant, beautiful, and highly readable type designs. A first version of Minion was released in 1990, Cyrillic characters were added in 1992 and a Multiple Master version was distributed soon afterwards. Finally, the present OpenType Minion Pro version was first released in 2000. The full Minion Pro family contains three weights and two widths, each with optical size variants, and each supporting a full range of Western languages, including Greek and Cyrillic. The fonts features many ligatures, small caps, oldstyle and proportional lining figures, superscripts and subscripts, ordinals and superior letters, swashes, alternates and ornaments. In general, the features that are supported in a OpenType font source are displayed with Eddie Kohler’s `otfinfo` program (see 11 for a list of features present in OpenType fonts). For instance, for the font *Minion Pro* the program displays:

```

otfinfo -f MinionPro-Regular.otf
aalt      Access All Alternates
c2sc      Small Capitals From Capitals
case      Case-Sensitive Forms
cpasp     Capital Spacing
dlig      Discretionary Ligatures
dnom      Denominators
fina      Terminal Forms
frac      Fractions
hist      Historical Forms
kern      Kerning
liga      Standard Ligatures
lnum      Lining Figures
numr      Numerators
onum      Oldstyle Figures
ordn      Ordinals

```

ornm	Ornaments
pnum	Proportional Figures
salt	Stylistic Alternates
sinf	Scientific Inferiors
size	Optical Size
smcp	Small Capitals
ss01	Stylistic Set 1
ss02	Stylistic Set 2
sup	Superscript
tnum	Tabular Figures
zero	Slashed Zero

The complete OpenType Minion font family comes in five packages (*Minion Pro* with 8 font instances, *Minion Pro Opticals* with 32 font instances, *Minion Pro Condensed* with 8 font instances, *Minion Pro Condensed Opticals* with 32 font instances, *Minion Std Black* with one font instance).<sup>1</sup> Rather than generating yourself all the  $\TeX$  support files for your Minion Pro font family, as explained in the previous sections, you can use the “ready-to-run” setup described next.

### 21.6.3.1 Installing the basic font files

The Minion Pro fonts can be used with  $\TeX$  with the help of the `minionpro` package developed by Achim Blumensath, Andreas Böhmann and Michael Zedler (CTAN:fonts/minionpro). In our case we have four instances of Minion Pro, which came with a recent version of Adobe’s *Acrobat Reader*.<sup>2</sup> As most people will have access to these fonts, we will document the setup for these four font instances only (for the other cases, refer to the package documentation). In this case you can download the file `base-v2.zip` and install the files therein by unpacking the zip archive into the root of the  $\TeX$  tree that contains the updates or local extensions to your  $\TeX$  system (e.g., `/TL2005/texmf-update`, `/TL2005/texmf-local`, or `c:\TL2005\localtexmf`). This will install a whole bunch of `.enc` (encoding), `.fd` (font definition), `.tfm` ( $\TeX$  font metrics), and `.vf` (virtual font) files. It also installs the fonts map `MinionPro.map` and the package files `fontaxes.sty`, `MinionPro-FontDef.sty`, `MinionPro.sty`, `mt-MinionPro.cfg`, and `otfontdef.sty`.

In order to use the OpenType font instances with  $\TeX$  on all systems it is easiest to transform the font into the PostScript Type 1 format. To achieve this the `minionpro` distribution comes with a set of scripts for performing this task on various platforms. Download the file `scripts.zip` and unpack it. Then refer to the `README` and follow its instructions. On Linux we run the `convert-lcdf` script, which for our four fonts amounts to the following:

```
> cfftot1 -b MinionPro-Regular.otf -o MinionPro-Regular.pfb
> tldotlessj MinionPro-Regular.pfb -o MinionPro-RegularLCDFJ.pfb
```

and similarly for the fonts `MinionPro-Bold`, `MinionPro-BoldIt`, and `MinionPro-It`. The `cfftot1` program (Section 21.4.1.1) generates a PostScript Type 1 font instance from the corresponding OpenType one. Its syntax follows.

```
cfftot1
Usage: cfftot1 [OPTIONS] [FONTFILE [OUTPUTFILE]]

Options:
    -a, --pfa                Output PFA font.
```

<sup>1</sup>Available from Adobe’s type Web site [http://store.adobe.com/type/browser/browser\\_M.html](http://store.adobe.com/type/browser/browser_M.html).

<sup>2</sup>On Linux systems the fonts for *Acrobat Reader* are usually in `/usr/lib/acroread/Resource/Font/`, on Microsoft Windows systems in `C:\Program Files\Adobe\Acrobat 7.0\Resource\Font`.

-b, --pfb	Output PFB font. This is the default.
-n, --name=NAME	Select font NAME from CFF.
-o, --output=FILE	Write output to FILE.
-q, --quiet	Do not generate any error messages.
-h, --help	Print this message and exit.
-v, --version	Print version number and exit.

The `t1dotlessj` program reads a standard PostScript Type 1 font (which does not have a dotless “j” character) and creates a new PostScript Type 1 font with just a dotlessj character, which is constructed by chopping the dot from the “j”. All .pfb files thus generated should be copied to the directory `texmf-local/fonts/type1/adobe/MinionPro` before updating the T<sub>E</sub>X database and declaring the map file, as follows.

```
[1]> texhash
[2]> updmap --enable Map MinionPro.map
```

Every glyph in a PostScript Type 1 font, including its name and encoding can be listed by the `t1testpage` program. Table 21.6 displays the layout of the *Minion Pro Regular* font generated with `t1testpage`.

Table 21.6: Minion Pro Regular font layout

A	A	a	a	A	a	B	B	b	B
b	C	c	c	C	D	D	d	D	d
E	E	e	e	E	e	F	F	f	F
G	G	g	g	G	H	H	h	H	h
I	I	i	i	I	J	J	j	J	K
K	K	k	k	K	L	L	l	L	l
M	M	m	m	M	N	N	n	N	n
N	n	O	o	O	o	P	P	p	P
P	Q	Q	q	Q	R	R	R	r	r
R	r	S	s	S	s	T	T	t	T
T	t	U	u	U	u	V	V	v	V
W	w	w	w	X	x	x	x	Y	Y
y	Y	Z	z	Z	z	(	(	(	(

(	(	.	.	.	.	.	.	.	.
,	,	-	-	-	-	-	-	-	-
;	;	?	?	)	)	)	)	)	)
0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	2	2	2
2	2	2	2	2	3	3	3	3	3
3	3	3	4	4	4	4	4	4	4
4	5	5	5	5	5	5	5	5	6
6	6	6	6	6	6	6	7	7	7
7	7	7	7	7	8	8	8	8	8
8	8	8	9	9	9	9	9	9	9
9	9	9	9	9	9	9	9	9	9
À	À	À	À	À	À	À	À	À	À
Á	Á	Á	Á	Á	Á	Á	Á	Á	Á

Latin letters, some digits

More digits, accented “a”

*Continued on next page*

[illegible]

Composites for “a,” “b,” “c,” “d”

[illegible]

Composites for “f,” “g,” “h,” “i,” “j,” “k,” “l”

Minion Pro font layout (*cont.*)[illegible][illegible]

f-ligatures, composites for “l,” “m,” “n,” “o”

greek_letters	bolding	diagonal	diagonal2	diagonal3	diagonal4	T_h	T_u	tu	Ts
α	α	α	α	α	α	Th	Tu	tu	Ts
Β	Β	Β	Β	Β	Β	Th	Tu	tu	Ts
Γ	Γ	Γ	Γ	Γ	Γ	Th	Tu	tu	Ts
Δ	Δ	Δ	Δ	Δ	Δ	Th	Tu	tu	Ts
Ε	Ε	Ε	Ε	Ε	Ε	Th	Tu	tu	Ts
Ζ	Ζ	Ζ	Ζ	Ζ	Ζ	Th	Tu	tu	Ts
Η	Η	Η	Η	Η	Η	Th	Tu	tu	Ts
Θ	Θ	Θ	Θ	Θ	Θ	Th	Tu	tu	Ts
Κ	Κ	Κ	Κ	Κ	Κ	Th	Tu	tu	Ts
Λ	Λ	Λ	Λ	Λ	Λ	Th	Tu	tu	Ts
Μ	Μ	Μ	Μ	Μ	Μ	Th	Tu	tu	Ts
Ν	Ν	Ν	Ν	Ν	Ν	Th	Tu	tu	Ts
Ξ	Ξ	Ξ	Ξ	Ξ	Ξ	Th	Tu	tu	Ts
Ο	Ο	Ο	Ο	Ο	Ο	Th	Tu	tu	Ts
Π	Π	Π	Π	Π	Π	Th	Tu	tu	Ts
Ρ	Ρ	Ρ	Ρ	Ρ	Ρ	Th	Tu	tu	Ts
Σ	Σ	Σ	Σ	Σ	Σ	Th	Tu	tu	Ts
Τ	Τ	Τ	Τ	Τ	Τ	Th	Tu	tu	Ts
Υ	Υ	Υ	Υ	Υ	Υ	Th	Tu	tu	Ts
Φ	Φ	Φ	Φ	Φ	Φ	Th	Tu	tu	Ts
Χ	Χ	Χ	Χ	Χ	Χ	Th	Tu	tu	Ts
Ψ	Ψ	Ψ	Ψ	Ψ	Ψ	Th	Tu	tu	Ts
Ω	Ω	Ω	Ω	Ω	Ω	Th	Tu	tu	Ts
Α	Α	Α	Α	Α	Α	Th	Tu	tu	Ts
Β	Β	Β	Β	Β	Β	Th	Tu	tu	Ts
Γ	Γ	Γ	Γ	Γ	Γ	Th	Tu	tu	Ts
Δ	Δ	Δ	Δ	Δ	Δ	Th	Tu	tu	Ts
Ε	Ε	Ε	Ε	Ε	Ε	Th	Tu	tu	Ts
Ζ	Ζ	Ζ	Ζ	Ζ	Ζ	Th	Tu	tu	Ts
Η	Η	Η	Η	Η	Η	Th	Tu	tu	Ts
Θ	Θ	Θ	Θ	Θ	Θ	Th	Tu	tu	Ts
Κ	Κ	Κ	Κ	Κ	Κ	Th	Tu	tu	Ts
Λ	Λ	Λ	Λ	Λ	Λ	Th	Tu	tu	Ts
Μ	Μ	Μ	Μ	Μ	Μ	Th	Tu	tu	Ts
Ν	Ν	Ν	Ν	Ν	Ν	Th	Tu	tu	Ts
Ξ	Ξ	Ξ	Ξ	Ξ	Ξ	Th	Tu	tu	Ts
Ο	Ο	Ο	Ο	Ο	Ο	Th	Tu	tu	Ts
Π	Π	Π	Π	Π	Π	Th	Tu	tu	Ts
Ρ	Ρ	Ρ	Ρ	Ρ	Ρ	Th	Tu	tu	Ts
Σ	Σ	Σ	Σ	Σ	Σ	Th	Tu	tu	Ts
Τ	Τ	Τ	Τ	Τ	Τ	Th	Tu	tu	Ts
Υ	Υ	Υ	Υ	Υ	Υ	Th	Tu	tu	Ts
Φ	Φ	Φ	Φ	Φ	Φ	Th	Tu	tu	Ts
Χ	Χ	Χ	Χ	Χ	Χ	Th	Tu	tu	Ts
Ψ	Ψ	Ψ	Ψ	Ψ	Ψ	Th	Tu	tu	Ts
Ω	Ω	Ω	Ω	Ω	Ω	Th	Tu	tu	Ts
α	α	α	α	α	α	Th	Tu	tu	Ts
β	β	β	β	β	β	Th	Tu	tu	Ts
γ	γ	γ	γ	γ	γ	Th	Tu	tu	Ts
δ	δ	δ	δ	δ	δ	Th	Tu	tu	Ts
ε	ε	ε	ε	ε	ε	Th	Tu	tu	Ts
ζ	ζ	ζ	ζ	ζ	ζ	Th	Tu	tu	Ts
η	η	η	η	η	η	Th	Tu	tu	Ts
θ	θ	θ	θ	θ	θ	Th	Tu	tu	Ts
κ	κ	κ	κ	κ	κ	Th	Tu	tu	Ts
λ	λ	λ	λ	λ	λ	Th	Tu	tu	Ts
μ	μ	μ	μ	μ	μ	Th	Tu	tu	Ts
ν	ν	ν	ν	ν	ν	Th	Tu	tu	Ts
ξ	ξ	ξ	ξ	ξ	ξ	Th	Tu	tu	Ts
ο	ο	ο	ο	ο	ο	Th	Tu	tu	Ts
π	π	π	π	π	π	Th	Tu	tu	Ts
ρ	ρ	ρ	ρ	ρ	ρ	Th	Tu	tu	Ts
σ	σ	σ	σ	σ	σ	Th	Tu	tu	Ts
τ	τ	τ	τ	τ	τ	Th	Tu	tu	Ts
υ	υ	υ	υ	υ	υ	Th	Tu	tu	Ts
φ	φ	φ	φ	φ	φ	Th	Tu	tu	Ts
χ	χ	χ	χ	χ	χ	Th	Tu	tu	Ts
ψ	ψ	ψ	ψ	ψ	ψ	Th	Tu	tu	Ts
ω	ω	ω	ω	ω	ω	Th	Tu	tu	Ts
Α	Α	Α	Α	Α	Α	Th	Tu	tu	Ts
Β	Β	Β	Β	Β	Β	Th	Tu	tu	Ts
Γ	Γ	Γ	Γ	Γ	Γ	Th	Tu	tu	Ts
Δ	Δ	Δ	Δ	Δ	Δ	Th	Tu	tu	Ts
Ε	Ε	Ε	Ε	Ε	Ε	Th	Tu	tu	Ts
Ζ	Ζ	Ζ	Ζ	Ζ	Ζ	Th	Tu	tu	Ts
Η	Η	Η	Η	Η	Η	Th	Tu	tu	Ts
Θ	Θ	Θ	Θ	Θ	Θ	Th	Tu	tu	Ts
Κ	Κ	Κ	Κ	Κ	Κ	Th	Tu	tu	Ts
Λ	Λ	Λ	Λ	Λ	Λ	Th	Tu	tu	Ts
Μ	Μ	Μ	Μ	Μ	Μ	Th	Tu	tu	Ts
Ν	Ν	Ν	Ν	Ν	Ν	Th	Tu	tu	Ts
Ξ	Ξ	Ξ	Ξ	Ξ	Ξ	Th	Tu	tu	Ts
Ο	Ο	Ο	Ο	Ο	Ο	Th	Tu	tu	Ts
Π	Π	Π	Π	Π	Π	Th	Tu	tu	Ts
Ρ	Ρ	Ρ	Ρ	Ρ	Ρ	Th	Tu	tu	Ts
Σ	Σ	Σ	Σ	Σ	Σ	Th	Tu	tu	Ts
Τ	Τ	Τ	Τ	Τ	Τ	Th	Tu	tu	Ts
Υ	Υ	Υ	Υ	Υ	Υ	Th	Tu	tu	Ts
Φ	Φ	Φ	Φ	Φ	Φ	Th	Tu	tu	Ts
Χ	Χ	Χ	Χ	Χ	Χ	Th	Tu	tu	Ts
Ψ	Ψ	Ψ	Ψ	Ψ	Ψ	Th	Tu	tu	Ts
Ω	Ω	Ω	Ω	Ω	Ω	Th	Tu	tu	Ts
Α	Α	Α	Α	Α	Α	Th	Tu	tu	Ts
Β	Β	Β	Β	Β	Β	Th	Tu	tu	Ts
Γ	Γ	Γ	Γ	Γ	Γ	Th	Tu	tu	Ts
Δ	Δ	Δ	Δ	Δ	Δ	Th	Tu	tu	Ts
Ε	Ε	Ε	Ε	Ε	Ε	Th	Tu	tu	Ts
Ζ	Ζ	Ζ	Ζ	Ζ	Ζ	Th	Tu	tu	Ts
Η	Η	Η	Η	Η	Η	Th	Tu	tu	Ts
Θ	Θ	Θ	Θ	Θ	Θ	Th	Tu	tu	Ts
Κ	Κ	Κ	Κ	Κ	Κ	Th	Tu	tu	Ts
Λ	Λ	Λ	Λ	Λ	Λ	Th	Tu	tu	Ts
Μ	Μ	Μ	Μ	Μ	Μ	Th	Tu	tu	Ts
Ν	Ν	Ν	Ν	Ν	Ν	Th	Tu	tu	Ts
Ξ	Ξ	Ξ	Ξ	Ξ	Ξ	Th	Tu	tu	Ts
Ο	Ο	Ο	Ο	Ο	Ο	Th	Tu	tu	Ts
Π	Π	Π	Π	Π	Π	Th	Tu	tu	Ts
Ρ	Ρ	Ρ	Ρ	Ρ	Ρ	Th	Tu	tu	Ts
Σ	Σ	Σ	Σ	Σ	Σ	Th	Tu	tu	Ts
Τ	Τ	Τ	Τ	Τ	Τ	Th	Tu	tu	Ts
Υ	Υ	Υ	Υ	Υ	Υ	Th	Tu	tu	Ts
Φ	Φ	Φ	Φ	Φ	Φ	Th	Tu	tu	Ts
Χ	Χ	Χ	Χ	Χ	Χ	Th	Tu	tu	Ts
Ψ	Ψ	Ψ	Ψ	Ψ	Ψ	Th	Tu	tu	Ts
Ω	Ω	Ω	Ω	Ω	Ω	Th	Tu	tu	Ts
Α	Α	Α	Α	Α	Α	Th	Tu	tu	Ts
Β	Β	Β	Β	Β	Β	Th	Tu	tu	Ts
Γ	Γ	Γ	Γ	Γ	Γ	Th	Tu	tu	Ts
Δ	Δ	Δ	Δ	Δ	Δ	Th	Tu	tu	Ts
Ε	Ε	Ε	Ε	Ε	Ε	Th	Tu	tu	Ts
Ζ	Ζ	Ζ	Ζ	Ζ	Ζ	Th	Tu	tu	Ts
Η	Η	Η	Η	Η	Η	Th	Tu	tu	Ts
Θ	Θ	Θ	Θ	Θ	Θ	Th	Tu	tu	Ts
Κ	Κ	Κ	Κ	Κ	Κ	Th	Tu	tu	Ts
Λ	Λ	Λ	Λ	Λ	Λ	Th	Tu	tu	Ts
Μ	Μ	Μ	Μ	Μ	Μ	Th	Tu	tu	Ts
Ν	Ν	Ν	Ν	Ν	Ν	Th	Tu	tu	Ts
Ξ	Ξ	Ξ	Ξ	Ξ	Ξ	Th	Tu	tu	Ts
Ο	Ο	Ο	Ο	Ο	Ο	Th	Tu	tu	Ts
Π	Π	Π	Π	Π	Π	Th	Tu	tu	Ts
Ρ	Ρ	Ρ	Ρ	Ρ	Ρ	Th	Tu	tu	Ts
Σ	Σ	Σ	Σ	Σ	Σ	Th	Tu	tu	Ts
Τ	Τ	Τ	Τ	Τ	Τ	Th	Tu	tu	Ts
Υ	Υ	Υ	Υ	Υ	Υ	Th	Tu	tu	Ts
Φ	Φ	Φ	Φ	Φ	Φ	Th	Tu	tu	Ts
Χ	Χ	Χ	Χ	Χ	Χ	Th	Tu	tu	Ts
Ψ	Ψ	Ψ	Ψ	Ψ	Ψ	Th	Tu	tu	Ts
Ω	Ω	Ω	Ω	Ω	Ω	Th	Tu	tu	Ts
Α	Α	Α	Α	Α	Α	Th	Tu	tu	Ts
Β	Β	Β	Β	Β	Β	Th	Tu	tu	Ts
Γ	Γ	Γ	Γ	Γ	Γ	Th	Tu	tu	Ts
Δ	Δ	Δ	Δ	Δ	Δ	Th	Tu	tu	Ts
Ε	Ε	Ε	Ε	Ε	Ε	Th	Tu	tu	Ts
Ζ	Ζ	Ζ	Ζ	Ζ	Ζ	Th	Tu	tu	Ts
Η	Η	Η	Η	Η	Η	Th	Tu	tu	Ts
Θ	Θ	Θ	Θ	Θ	Θ	Th	Tu	tu	Ts
Κ	Κ	Κ	Κ	Κ	Κ	Th	Tu	tu	Ts
Λ	Λ	Λ	Λ	Λ	Λ	Th	Tu	tu	Ts
Μ	Μ	Μ	Μ	Μ	Μ	Th	Tu	tu	Ts
Ν	Ν	Ν	Ν	Ν	Ν	Th	Tu	tu	Ts
Ξ	Ξ	Ξ	Ξ	Ξ	Ξ	Th	Tu	tu	Ts
Ο	Ο	Ο	Ο	Ο	Ο	Th	Tu	tu	Ts
Π	Π	Π	Π	Π	Π	Th	Tu	tu	Ts
Ρ	Ρ	Ρ	Ρ	Ρ	Ρ	Th	Tu	tu	Ts
Σ	Σ	Σ	Σ	Σ	Σ	Th	Tu	tu	Ts
Τ	Τ	Τ	Τ	Τ	Τ	Th	Tu	tu	Ts
Υ	Υ	Υ	Υ	Υ	Υ	Th	Tu	tu	Ts
Φ	Φ	Φ	Φ	Φ	Φ	Th	Tu	tu	Ts
Χ	Χ	Χ	Χ	Χ	Χ	Th	Tu	tu	Ts
Ψ	Ψ	Ψ	Ψ	Ψ	Ψ	Th	Tu	tu	Ts
Ω	Ω	Ω	Ω	Ω	Ω	Th	Tu	tu	Ts
Α	Α	Α	Α	Α	Α	Th	Tu	tu	Ts
Β	Β	Β	Β	Β	Β	Th	Tu	tu	Ts
Γ	Γ	Γ	Γ	Γ	Γ	Th	Tu	tu	Ts
Δ	Δ	Δ	Δ	Δ	Δ	Th	Tu	tu	Ts
Ε	Ε	Ε	Ε	Ε	Ε	Th	Tu	tu	Ts
Ζ	Ζ	Ζ	Ζ	Ζ	Ζ	Th	Tu	tu	Ts
Η	Η	Η	Η	Η	Η	Th	Tu	tu	Ts
Θ	Θ	Θ	Θ	Θ	Θ	Th	Tu	tu	Ts
Κ	Κ	Κ	Κ	Κ	Κ	Th	Tu	tu	Ts
Λ	Λ	Λ	Λ	Λ	Λ	Th	Tu	tu	Ts
Μ	Μ	Μ	Μ	Μ	Μ	Th	Tu	tu	Ts
Ν	Ν	Ν	Ν	Ν	Ν	Th	Tu	tu	Ts
Ξ	Ξ	Ξ	Ξ	Ξ	Ξ	Th	Tu	tu	Ts
Ο	Ο	Ο	Ο	Ο	Ο	Th	Tu	tu	Ts
Π	Π	Π	Π	Π	Π	Th	Tu	tu	Ts
Ρ	Ρ	Ρ	Ρ	Ρ	Ρ	Th	Tu	tu	Ts
Σ	Σ	Σ	Σ	Σ	Σ	Th	Tu	tu	Ts
Τ	Τ	Τ	Τ	Τ	Τ	Th	Tu	tu	Ts
Υ	Υ	Υ	Υ	Υ	Υ	Th	Tu	tu	Ts
Φ	Φ	Φ	Φ	Φ	Φ	Th	Tu	tu	Ts
Χ	Χ	Χ	Χ	Χ	Χ	Th	Tu	tu	Ts
Ψ	Ψ	Ψ	Ψ	Ψ	Ψ	Th	Tu	tu	Ts
Ω	Ω	Ω	Ω	Ω	Ω	Th	Tu	tu	Ts
Α	Α	Α	Α	Α	Α	Th	Tu	tu	Ts
Β	Β	Β	Β	Β	Β	Th	Tu	tu	Ts
Γ	Γ	Γ	Γ	Γ	Γ	Th	Tu	tu	Ts
Δ	Δ	Δ	Δ	Δ	Δ	Th	Tu	tu	Ts
Ε	Ε	Ε	Ε	Ε	Ε	Th	Tu	tu	Ts
Ζ	Ζ	Ζ	Ζ	Ζ	Ζ	Th	Tu	tu	Ts
Η	Η	Η	Η	Η	Η	Th	Tu	tu	Ts
Θ	Θ	Θ	Θ	Θ	Θ	Th</			

Pi characters, composites for "o," "p," "r," "s"

unitedAc	unitedB	unitedC	unitedD	unitedEand	unitedF	unitedG	unitedH	unitedI	unitedJand
Ā	Ē	Ė	ē	ĕ	Ě	Ė	ĕ	Ė	Ė
unitedK	unitedL	unitedM	unitedN	unitedOand	unitedP	unitedQ	unitedR	unitedS	unitedTand
Ē	Ē	Ē	Ē	Ē	Ē	Ē	Ē	Ē	Ē
unitedU	unitedV	unitedW	unitedX	unitedYand	unitedZ	unitedAA	unitedAB	unitedAC	unitedADand
Ē	Ē	Ē	Ē	Ē	Ē	Ē	Ē	Ē	Ē
unitedAE	unitedAF	unitedAG	unitedAH	unitedAIand	unitedAJ	unitedAK	unitedAL	unitedAM	unitedANand
Ē	Ē	Ē	Ē	Ē	Ē	Ē	Ē	Ē	Ē
unitedAO	unitedAP	unitedAQ	unitedAR	unitedASand	unitedAT	unitedAU	unitedAV	unitedAW	unitedAXand
Ē	Ē	Ē	Ē	Ē	Ē	Ē	Ē	Ē	Ē
unitedAY	unitedBz	unitedBz	unitedCz	unitedDz	unitedEz	unitedFz	unitedGz	unitedHz	unitedIz
Ē	Ē	Ē	Ē	Ē	Ē	Ē	Ē	Ē	Ē
unitedJz	unitedKz	unitedLz	unitedMz	unitedNz	unitedOz	unitedPz	unitedQz	unitedRz	unitedS
Ē	Ē	Ē	Ē	Ē	Ē	Ē	Ē	Ē	Ē
unitedTz	unitedUz	unitedVz	unitedWz	unitedXz	unitedYz	unitedZz	unitedAAz	unitedABz	unitedC
Ē	Ē	Ē	Ē	Ē	Ē	Ē	Ē	Ē	Ē
unitedDz	unitedEz	unitedFz	unitedGz	unitedHz	unitedIz	unitedJz	unitedKz	unitedLz	unitedM
Ē	Ē	Ē	Ē	Ē	Ē	Ē	Ē	Ē	Ē
unitedNz	unitedOz	unitedPz	unitedQz	unitedRz	unitedSz	unitedTz	unitedUz	unitedVz	unitedW
Ē	Ē	Ē	Ē	Ē	Ē	Ē	Ē	Ē	Ē
unitedXz	unitedYz	unitedZz	unitedAAz	unitedABz	unitedCz	unitedDz	unitedEz	unitedFz	unitedG
Ē	Ē	Ē	Ē	Ē	Ē	Ē	Ē	Ē	Ē
unitedHz	unitedIz	unitedJz	unitedKz	unitedLz	unitedMz	unitedNz	unitedOz	unitedPz	unitedQ
Ē	Ē	Ē	Ē	Ē	Ē	Ē	Ē	Ē	Ē
unitedRz	unitedSz	unitedTz	unitedUz	unitedVz	unitedWz	unitedXz	unitedYz	unitedZz	unitedA
Ē	Ē	Ē	Ē	Ē	Ē	Ē	Ē	Ē	Ē
unitedBz	unitedCz	unitedDz	unitedEz	unitedFz	unitedGz	unitedHz	unitedIz	unitedJz	unitedK
Ē	Ē	Ē	Ē	Ē	Ē	Ē	Ē	Ē	Ē
unitedLz	unitedMz	unitedNz	unitedOz	unitedPz	unitedQz	unitedRz	unitedSz	unitedTz	unitedU
Ē	Ē	Ē	Ē	Ē	Ē	Ē	Ē	Ē	Ē
unitedVz	unitedWz	unitedXz	unitedYz	unitedZz	unitedAAz	unitedABz	unitedCz	unitedDz	unitedE
Ē	Ē	Ē	Ē	Ē	Ē	Ē	Ē	Ē	Ē
unitedFz	unitedGz	unitedHz	unitedIz	unitedJz	unitedKz	unitedLz	unitedMz	unitedNz	unitedO
Ē	Ē	Ē	Ē	Ē	Ē	Ē	Ē	Ē	Ē
unitedPz	unitedQz	unitedRz	unitedSz	unitedTz	unitedUz	unitedVz	unitedWz	unitedXz	unitedY
Ē	Ē	Ē	Ē	Ē	Ē	Ē	Ē	Ē	Ē
unitedZz	unitedAAz	unitedABz	unitedCz	unitedDz	unitedEz	unitedFz	unitedGz	unitedHz	unitedI
Ē	Ē	Ē	Ē	Ē	Ē	Ē	Ē	Ē	Ē
unitedJz	unitedKz	unitedLz	unitedMz	unitedNz	unitedOz	unitedPz	unitedQz	unitedRz	unitedS
Ē	Ē	Ē	Ē	Ē	Ē	Ē	Ē	Ē	Ē
unitedTz	unitedUz	unitedVz	unitedWz	unitedXz	unitedYz	unitedZz	unitedAAz	unitedABz	unitedC
Ē	Ē	Ē	Ē	Ē	Ē	Ē	Ē	Ē	Ē
unitedDz	unitedEz	unitedFz	unitedGz	unitedHz	unitedIz	unitedJz	unitedKz	unitedLz	unitedM
Ē	Ē	Ē	Ē	Ē	Ē	Ē	Ē	Ē	Ē
unitedNz	unitedOz	unitedPz	unitedQz	unitedRz	unitedSz	unitedTz	unitedUz	unitedVz	unitedW
Ē	Ē	Ē	Ē	Ē	Ē	Ē	Ē	Ē	Ē
unitedXz	unitedYz	unitedZz	unitedAAz	unitedABz	unitedCz	unitedDz	unitedEz	unitedFz	unitedG
Ē	Ē	Ē	Ē	Ē	Ē	Ē	Ē	Ē	Ē
unitedHz	unitedIz	unitedJz	unitedKz	unitedLz	unitedMz	unitedNz	unitedOz	unitedPz	unitedQ
Ē	Ē	Ē	Ē	Ē	Ē	Ē	Ē	Ē	Ē

Composites for “t,” “u,” doubly accented “a”

Doubly accented “e,” “i,” “o,” “s,” “y”

*Continued on next page*



un19f4c	un19f7	un19f7a	un19f8	un19f8c	un19f8d	un19f8e	un19f8f	un19f8g	un19f8h	un19f8i	un19f8j	un19f8k	un19f8l	un19f8m	un19f8n	un19f8o	un19f8p	un19f8q	un19f8r	un19f8s	un19f8t	un19f8u	un19f8v	un19f8w	un19f8x	un19f8y	un19f8z	un19f8aa	un19f8ab	un19f8ac	un19f8ad	un19f8ae	un19f8af	un19f8ag	un19f8ah	un19f8ai	un19f8aj	un19f8ak	un19f8al	un19f8am	un19f8an	un19f8ao	un19f8ap	un19f8aq	un19f8ar	un19f8as	un19f8at	un19f8au	un19f8av	un19f8aw	un19f8ax	un19f8ay	un19f8az	un19f8ba	un19f8bb	un19f8bc	un19f8bd	un19f8be	un19f8bf	un19f8bg	un19f8bh	un19f8bi	un19f8bj	un19f8bk	un19f8bl	un19f8bm	un19f8bn	un19f8bo	un19f8bp	un19f8bq	un19f8br	un19f8bs	un19f8bt	un19f8bu	un19f8bv	un19f8bw	un19f8bx	un19f8by	un19f8bz	un19f8ca	un19f8cb	un19f8cc	un19f8cd	un19f8ce	un19f8cf	un19f8cg	un19f8ch	un19f8ci	un19f8cj	un19f8ck	un19f8cl	un19f8cm	un19f8cn	un19f8co	un19f8cp	un19f8cq	un19f8cr	un19f8cs	un19f8ct	un19f8cu	un19f8cv	un19f8cw	un19f8cx	un19f8cy	un19f8cz	un19f8da	un19f8db	un19f8dc	un19f8dd	un19f8de	un19f8df	un19f8dg	un19f8dh	un19f8di	un19f8dj	un19f8dk	un19f8dl	un19f8dm	un19f8dn	un19f8do	un19f8dp	un19f8dq	un19f8dr	un19f8ds	un19f8dt	un19f8du	un19f8dv	un19f8dw	un19f8dx	un19f8dy	un19f8dz	un19f8ea	un19f8eb	un19f8ec	un19f8ed	un19f8ee	un19f8ef	un19f8eg	un19f8eh	un19f8ei	un19f8ej	un19f8ek	un19f8el	un19f8em	un19f8en	un19f8eo	un19f8ep	un19f8eq	un19f8er	un19f8es	un19f8et	un19f8eu	un19f8ev	un19f8ew	un19f8ex	un19f8ey	un19f8ez	un19f8fa	un19f8fb	un19f8fc	un19f8fd	un19f8fe	un19f8ff	un19f8fg	un19f8fh	un19f8fi	un19f8fj	un19f8fk	un19f8fl	un19f8fm	un19f8fn	un19f8fo	un19f8fp	un19f8fq	un19f8fr	un19f8fs	un19f8ft	un19f8fu	un19f8fv	un19f8fw	un19f8fx	un19f8fy	un19f8fz	un19f8ga	un19f8gb	un19f8gc	un19f8gd	un19f8ge	un19f8gf	un19f8gg	un19f8gh	un19f8gi	un19f8gj	un19f8gk	un19f8gl	un19f8gm	un19f8gn	un19f8go	un19f8gp	un19f8gq	un19f8gr	un19f8gs	un19f8gt	un19f8gu	un19f8gv	un19f8gw	un19f8gx	un19f8gy	un19f8gz	un19f8ha	un19f8hb	un19f8hc	un19f8hd	un19f8he	un19f8hf	un19f8hg	un19f8hi	un19f8hj	un19f8hk	un19f8hl	un19f8hm	un19f8hn	un19f8ho	un19f8hp	un19f8hq	un19f8hr	un19f8hs	un19f8ht	un19f8hu	un19f8hv	un19f8hw	un19f8hx	un19f8hy	un19f8hz	un19f8ia	un19f8ib	un19f8ic	un19f8id	un19f8ie	un19f8if	un19f8ig	un19f8ih	un19f8ii	un19f8ij	un19f8ik	un19f8il	un19f8im	un19f8in	un19f8io	un19f8ip	un19f8iq	un19f8ir	un19f8is	un19f8it	un19f8iu	un19f8iv	un19f8iw	un19f8ix	un19f8iy	un19f8iz	un19f8ja	un19f8jb	un19f8jc	un19f8jd	un19f8je	un19f8jf	un19f8jg	un19f8jh	un19f8ji	un19f8jj	un19f8jk	un19f8jl	un19f8jm	un19f8jn	un19f8jo	un19f8jp	un19f8jq	un19f8jr	un19f8js	un19f8jt	un19f8ju	un19f8jv	un19f8jw	un19f8jx	un19f8jy	un19f8jz	un19f8ka	un19f8kb	un19f8kc	un19f8kd	un19f8ke	un19f8kf	un19f8kg	un19f8kh	un19f8ki	un19f8kj	un19f8kk	un19f8kl	un19f8km	un19f8kn	un19f8ko	un19f8kp	un19f8kq	un19f8kr	un19f8ks	un19f8kt	un19f8ku	un19f8kv	un19f8kw	un19f8kx	un19f8ky	un19f8kz	un19f8la	un19f8lb	un19f8lc	un19f8ld	un19f8le	un19f8lf	un19f8lg	un19f8lh	un19f8li	un19f8lj	un19f8lk	un19f8ll	un19f8lm	un19f8ln	un19f8lo	un19f8lp	un19f8lq	un19f8lr	un19f8ls	un19f8lt	un19f8lu	un19f8lv	un19f8lw	un19f8lx	un19f8ly	un19f8lz	un19f8ma	un19f8mb	un19f8mc	un19f8md	un19f8me	un19f8mf	un19f8mg	un19f8mh	un19f8mi	un19f8mj	un19f8mk	un19f8ml	un19f8mm	un19f8mn	un19f8mo	un19f8mp	un19f8mq	un19f8mr	un19f8ms	un19f8mt	un19f8mu	un19f8mv	un19f8mw	un19f8mx	un19f8my	un19f8mz	un19f8na	un19f8nb	un19f8nc	un19f8nd	un19f8ne	un19f8nf	un19f8ng	un19f8nh	un19f8ni	un19f8nj	un19f8nk	un19f8nl	un19f8nm	un19f8nn	un19f8no	un19f8np	un19f8nq	un19f8nr	un19f8ns	un19f8nt	un19f8nu	un19f8nv	un19f8nw	un19f8nx	un19f8ny	un19f8nz	un19f8oa	un19f8ob	un19f8oc	un19f8od	un19f8oe	un19f8of	un19f8og	un19f8oh	un19f8oi	un19f8oj	un19f8ok	un19f8ol	un19f8om	un19f8on	un19f8oo	un19f8op	un19f8oq	un19f8or	un19f8os	un19f8ot	un19f8ou	un19f8ov	un19f8ow	un19f8ox	un19f8oy	un19f8oz	un19f8pa	un19f8pb	un19f8pc	un19f8pd	un19f8pe	un19f8pf	un19f8pg	un19f8ph	un19f8pi	un19f8pj	un19f8pk	un19f8pl	un19f8pm	un19f8pn	un19f8po	un19f8pp	un19f8pq	un19f8pr	un19f8ps	un19f8pt	un19f8pu	un19f8pv	un19f8pw	un19f8px	un19f8py	un19f8pz	un19f8qa	un19f8qb	un19f8qc	un19f8qd	un19f8qe	un19f8qf	un19f8qg	un19f8qh	un19f8qi	un19f8qj	un19f8qk	un19f8ql	un19f8qm	un19f8qn	un19f8qo	un19f8qp	un19f8qq	un19f8qr	un19f8qs	un19f8qt	un19f8qu	un19f8qv	un19f8qw	un19f8qx	un19f8qy	un19f8qz	un19f8ra	un19f8rb	un19f8rc	un19f8rd	un19f8re	un19f8rf	un19f8rg	un19f8rh	un19f8ri	un19f8rj	un19f8rk	un19f8rl	un19f8rm	un19f8rn	un19f8ro	un19f8rp	un19f8rq	un19f8rr	un19f8rs	un19f8rt	un19f8ru	un19f8rv	un19f8rw	un19f8rx	un19f8ry	un19f8rz	un19f8sa	un19f8sb	un19f8sc	un19f8sd	un19f8se	un19f8sf	un19f8sg	un19f8sh	un19f8si	un19f8sj	un19f8sk	un19f8sl	un19f8sm	un19f8sn	un19f8so	un19f8sp	un19f8sq	un19f8sr	un19f8ss	un19f8st	un19f8su	un19f8sv	un19f8sw	un19f8sx	un19f8sy	un19f8sz	un19f8ta	un19f8tb	un19f8tc	un19f8td	un19f8te	un19f8tf	un19f8tg	un19f8th	un19f8ti	un19f8tj	un19f8tk	un19f8tl	un19f8tm	un19f8tn	un19f8to	un19f8tp	un19f8tq	un19f8tr	un19f8ts	un19f8tt	un19f8tu	un19f8tv	un19f8tw	un19f8tx	un19f8ty	un19f8tz	un19f8ua	un19f8ub	un19f8uc	un19f8ud	un19f8ue	un19f8uf	un19f8ug	un19f8uh	un19f8ui	un19f8uj	un19f8uk	un19f8ul	un19f8um	un19f8un	un19f8uo	un19f8up	un19f8uq	un19f8ur	un19f8us	un19f8ut	un19f8uu	un19f8uv	un19f8uw	un19f8ux	un19f8uy	un19f8uz	un19f8va	un19f8vb	un19f8vc	un19f8vd	un19f8ve	un19f8vf	un19f8vg	un19f8vh	un19f8vi	un19f8vj	un19f8vk	un19f8vl	un19f8vm	un19f8vn	un19f8vo	un19f8vp	un19f8vq	un19f8vr	un19f8vs	un19f8vt	un19f8vu	un19f8vv	un19f8vw	un19f8vx	un19f8vy	un19f8vz	un19f8wa	un19f8wb	un19f8wc	un19f8wd	un19f8we	un19f8wf	un19f8wg	un19f8wh	un19f8wi	un19f8wj	un19f8wk	un19f8wl	un19f8wm	un19f8wn	un19f8wo	un19f8wp	un19f8wq	un19f8wr	un19f8ws	un19f8wt	un19f8wu	un19f8wv	un19f8ww	un19f8wx	un19f8wy	un19f8wz	un19f8xa	un19f8xb	un19f8xc	un19f8xd	un19f8xe	un19f8xf	un19f8xg	un19f8xh	un19f8xi	un19f8xj	un19f8xk	un19f8xl	un19f8xm	un19f8xn	un19f8xo	un19f8xp	un19f8xq	un19f8xr	un19f8xs	un19f8xt	un19f8xu	un19f8xv	un19f8xw	un19f8xx	un19f8xy	un19f8xz	un19f8ya	un19f8yb	un19f8yc	un19f8yd	un19f8ye	un19f8yf	un19f8yg	un19f8yh	un19f8yi	un19f8yj	un19f8yk	un19f8yl	un19f8ym	un19f8yn	un19f8yo	un19f8yp	un19f8yq	un19f8yr	un19f8ys	un19f8yt	un19f8yu	un19f8yv	un19f8yw	un19f8yx	un19f8yy	un19f8yz	un19f8za	un19f8zb	un19f8zc	un19f8zd	un19f8ze	un19f8zf	un19f8zg	un19f8zh	un19f8zi	un19f8zj	un19f8zk	un19f8zl	un19f8zm	un19f8zn	un19f8zo	un19f8zp	un19f8zq	un19f8zr	un19f8zs	un19f8zt	un19f8zu	un19f8zv	un19f8zw	un19f8zx	un19f8zy	un19f8zz
---------	--------	---------	--------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------

[illegible]

Achim Blumensath has also developed a font MnSymbol (CTAN:fonts/mnsymbol) with a large set of mathematical symbols as a companion to Adobe MinionPro. This font can be used with  $\text{\LaTeX}$  by loading the MnSymbol package.

The MnSymbol package automatically loads the amsmath and textcomp packages. It is incompatible

default	0123456789		
text	0123456789	lining	0123456789
text,proportional	0123456789	lining, proportional	0123456789
text,proportional	1114448880	lining, proportional	1114448880
text,tabular	0123456789	lining,tabular	0123456789
text,tabular	1114448880	lining,tabular	1114448880

Figure 21.21: Types of figures with Minion Pro

with the `amssymb` and `amsfonts` packages. `MnSymbol` offers the following options:

`mnsy` calligraphic font is taken from `MnSymbol` (default);  
`cmsy` calligraphic font is taken from Computer Modern;  
`abx` calligraphic font is taken from the (experimental) `mathabx` fonts.

### 21.6.3.3 Running with the Minion Pro fonts

The `MinionPro` package is loaded by placing the following commands in your document preamble.

```
\usepackage{MinionPro}
\usepackage{MnSymbol}
```

The main document options for `MinionPro` are described next.

#### 21.6.3.4 – Figure selection

`MinionPro` offers four different figure versions: *text* (lowercase) figures or *lining* (uppercase) figures, and *proportional* (different widths) or *tabular* (same width) figures, the latter useful in table alignments.

The `\figureversion` command lets you switch between these different figure versions in the text. The command takes the following parameters:

**text, osf** text figures;  
**lining, lf** lining figures;  
**tabular, tab** tabular figures;  
**proportional, prop** proportional figures.

Figure 21.21 shows various combinations of these parameters. It is seen that `proportional` digits (the default) do not align properly, hence inside tables one should always use the `tabular` version.

The figure version to be used as the default for the document is set by selecting one of the following options.

`textosf` use text figures in text mode;  
`mathosf` use text figures in math mode;  
`osf` use text figures in text and math mode (default);  
`textlf` use lining figures in text mode;  
`mathlf` use lining figures in math mode;  
`lf` use lining figures in text and math mode;  
`mathtabular` use tabular figures in math mode.

Exa.  
21-6-2




















































100		101		102		103		104		105		106		107		108		109	
110		111		112		113		114		115		116		117		118		119	
120		121		122		123		124		125		126		127		128		129	
130		131		132		133		134		135		136		137		138		139	
140		141		142		143		144		145		146		147		148		149	
150																			

Table 21.7: The Pi in the PostScript font Minion Pro

### 21.6.3.5 – Calligraphic fonts

The options available to specify which font will be used by the `\mathcal` command are those of the `MnSymbol` package (see above).

### 21.6.3.6 – Blackboard bold letters

The font to be used by the `\mathbb` command is selected as follows.

`amsbb`            use the AMS blackboard font (default);  
`fourierbb`       use the Fourier blackboard font;  
`lucidabb`        use the (commercial) Lucida Math blackboard font.

### 21.6.3.7 – Greek letters

Options to specify whether you want to use upright or italic Greek letters in math mode. are:

`mixedgreek`     uppercase Greek is upright, lowercase Greek is italic (default);  
`italicgreek`     all Greek letters are italic;  
`frenchmath`      all Greek letters and the uppercase Roman letters are upright.

Upright and italic Greek letters are also directly accessible via the commands `\upgamma`, `\itgamma`, `\upGamma`, `\itGamma`, etc.

### 21.6.3.8 – Special symbol selection

`minionint`            the integral symbols are taken from Minion Pro rather than from `MnSymbol`;  
`footnotefigures`    use special figures for footnote marks.

### 21.6.3.9 Additional font shapes, symbols and ornaments

The Minion Pro font contains many additional characters. For instance, in addition to the normal small caps shape (`\scshape` and `\textsc`) there is a letterspaced version (`\sscshape` and `\textssc`). There also are swash capitals (`\swshape` and `\textsw`). These commands are shown in action in the following example.

Exa.  
21-6-3

Text in the normal text font.

TEXT IN SMALL CAPITALS.

TEXT IN EXTENDED SMALL CAPITALS.

*Text with swash capitals.*

```
\usepackage{MinionPro}
      Text in the normal text font.\\
\textsc{Text in small capitals.}\\
\textssc{Text in extended small capitals.}\\
\textsw{Text with swash capitals.}\\
```

Latin: OT1, T1, TS1, LY1, T5  
Cyrillic: T2A, T2B, T2C, X2, OT2  
Greek: LGR (to be used with babel, including polutonikogreek)

Table 21.8: Encoding support by language in the MinionPro package

encoding	family	series	shape
OT1, T1, TS1, LY1, T5	MinionPro-OsF, MinionPro-LF, MinionPro-TOf, MinionPro-TLF	m, b (sb, bx), eb	n, it (sl), sw1, sc, scit (scsl, scsw), ssc, sscit (sscs, sscsw)
lgr, lgi, t2a, t2b, t2c, x2, ot2	MinionPro-OsF, MinionPro-LF, MinionPro-TOf, MinionPro-TLF	m, b (sb, bx), eb	n, it (sl)
oml, omlfrench	MinionPro-TOf	m, b (sb, bx), eb	n, it
u	MinionPro-Extra	m, b (sb, bx), eb	n, it (sl)

Font instances between parentheses are provided via  $\TeX$ 's font substitution mechanism

Table 21.9: Font selections available for various encodings in the MinionPro package

There exist also small and slanted fractions, which are constructed with the `\smallfrac` and `\slantfrac` commands, which have two arguments, the numerator and the denominator (these can only be positive integers), as shown in the next example.

$\frac{1}{4}\frac{7}{23}$  and  $\frac{1}{4}\frac{7}{23}$   
 $\sin(\frac{1}{5}a + \frac{3}{7}b)$   
We drank a  $\frac{1}{4}$  liter glass of vodka.

```
\usepackage{MinionPro}  
\smallfrac{1}{4} \smallfrac{7}{23} and  
\slantfrac{1}{4} \slantfrac{7}{23}\  
$\sin(\smallfrac{1}{5}a+\smallfrac{3}{7}b)$\  
We drank a \slantfrac{1}{4} liter glass of vodka.
```

Exa.  
21-6-4

Ornaments (Pi characters) can be accents with the help of the `pifont` package, which is described in detail in Section 21.3.2. Table 21.7 on the preceding page shows the symbols that are available together with their reference number. As an example we show a variant of Example 21-3-6 using Pi symbols of the Minion Pro font.

☞ ☞ ☞ ☞ ☞ ☞ ☞ ☞  
☞ ☞ ☞ ☞ ☞ text ☞ text ☞ ☞ ☞ ☞ ☞

```
\usepackage{pifont}  
\Piline{MinionPro-Extra}{113} \par\medskip  
\noindent\Pifill{MinionPro-Extra}{101} text  
\Pisymbol{MinionPro-Extra}{102}  
text\Pifill{MinionPro-Extra}{103}
```

Exa.  
21-6-5

21.6.3.10 Language and font support

Tables 21.8 and 21.9 show, respectively, how the MinionPro package supports the main encodings per language, and which font selections (family, series, shape) are available for the various encodings.

Table 21.10: The standard PostScript fonts in the *Fontname* scheme.

pagk	AvantGarde-Book	phvrrn	Helvetica-Narrow
pagko	AvantGarde-BookOblique	phvron	Helvetica-NarrowOblique
pagd	AvantGarde-Demi	pncb	NewCenturySchlbk-Bold
pagdo	AvantGarde-DemiOblique	pncbi	NewCenturySchlbk-BoldItalic
pbkd	Bookman-Demi	pncr	NewCenturySchlbk-Italic
pbkdi	Bookman-DemiItalic	pncr	NewCenturySchlbk-Roman
pbkl	Bookman-Light	pplb	Palatino-Bold
pbkli	Bookman-LightItalic	pplbi	Palatino-BoldItalic
pcrb	Courier-Bold	pplri	Palatino-Italic
pcrbo	Courier-BoldOblique	pplr	Palatino-Roman
pcrr	Courier	psyr	Symbol
pcrro	Courier-Oblique	ptmb	Times-Bold
phvb	Helvetica-Bold	ptmbi	Times-BoldItalic
phvbo	Helvetica-BoldOblique	ptmri	Times-Italic
phvbrn	Helvetica-NarrowBold	ptmr	Times-Roman
phvbob	Helvetica-NarrowBoldOblique	pzcmi	ZapfChancery-MediumItalic
phvr	Helvetica	pzdr	ZapfDingbats
phvro	Helvetica-Oblique		

### 21.6.3.11 A complete example

Figure 21.22 on the following page shows side-by-side a short example containing math and text typeset with the Minion Pro font. As the font contains both Cyrillic and Latin characters the visual compatibility in appearance in both alphabets is ideally guaranteed. For the top view we used the default option settings of the MinionPro package, whereas for the bottom view we used with the same source file but with the following option settings:

- frenchmath** which typesets all Greek letters upright. The default (upper part of the figure) is that lowercase Greek letters are typeset in italic, and uppercase Greek upright (compare in particular the line containing  $\alpha, \beta, \gamma, \dots$ ).
- lf** Lining (uppercase) digits are used instead of the text (lowercase) default representations (compare the heights of the digits in the upper and lower part of Figure 21.22 on the next page).
- minionint** Use the integral sign of Minion Pro instead of that in the companion font MnSymbol.

## 21.7 Classifying PostScript fonts

This section describes Version 2.0 of *Fontname*, a naming scheme for  $\text{\TeX}$  font filenames that is the basis of the  $\text{\LaTeX}$  PSNFSS support for PostScript fonts. *Fontname* is maintained by Karl Berry.<sup>1</sup> The aim of *Fontname* is to define short, portable filenames. To ensure portability across the maximum number of platforms, filenames are limited to eight characters, for compatibility with DOS filesystems (now only of historical interest) and the ISO 9660 standard used for CD-ROM distribution. Moreover, filenames consist only of letters (monocase), numerals, and underscore. A typical example is shown in Table 21.10, which lists short names for the standard 35 PostScript fonts.

<sup>1</sup>The full documentation is available as an electronic document on CTAN at: [info/fontname](http://info/fontname).

## 1 Общие сведения

Заметьте разницу в стиле верстки выражений в абзацах и выключных:  $\lim_{n \rightarrow \infty} \sum_{k=1}^n \frac{1}{k^2} = \frac{\pi^2}{6} = 1.644934$ :

$$\lim_{n \rightarrow \infty} \sum_{k=1}^n \frac{1}{k^2} = \frac{\pi^2}{6} = 1.644934. \quad (1)$$

**Строчные греческие буквы** вводятся как `\alpha`, `\beta`, `\gamma`, ..., **прописные буквы** вводятся как `\Gamma`, `\Delta`, ...,  $\alpha$ ,  $\beta$ ,  $\gamma$ ,  $\lambda$ ,  $\xi$ ,  $\pi$ ,  $\mu$ ,  $\Phi$ ,  $\Omega$ .

**Оператор интеграла** печатает команда `\int` ( $\int_0^\pi$ ).

$$\int_0^\pi \sqrt{\pi^2 - x^2} dx = \frac{\pi^3}{4} \quad (2)$$

Окружение `array` можно также использовать для верстки выражений, имеющих один большой ограничитель, подставляя `\right` в качестве невидимого правого ограничителя:

$$y = \begin{cases} +1 & \text{если } d > c, \\ -1 & \text{по утрам,} \\ 0 & \text{остальное время дня.} \end{cases}$$

## 2 Généralités

Remarquez le format différent des expressions mathématiques composées « en ligne » ou « hors texte » :  $\lim_{n \rightarrow \infty} \sum_{k=1}^n \frac{1}{k^2} = \frac{\pi^2}{6} = 1.644934$ ,

$$\lim_{n \rightarrow \infty} \sum_{k=1}^n \frac{1}{k^2} = \frac{\pi^2}{6} = 1.644934. \quad (3)$$

Les lettres **grecques minuscules** sont saisies de la manière suivante : `\alpha`, `\beta`, `\gamma`, etc. Les lettres **grecques majuscules** sont saisies ainsi : `\Gamma`, `\Delta`, etc. :  $\alpha$ ,  $\beta$ ,  $\gamma$ ,  $\lambda$ ,  $\xi$ ,  $\pi$ ,  $\mu$ ,  $\Phi$ ,  $\Omega$ .

La commande `\int` produit une **intégrale** ( $\int_0^\pi$ ).

$$\int_0^\pi \sqrt{\pi^2 - x^2} dx = \frac{\pi^3}{4} \quad (4)$$

L'environnement `array` peut également être utilisé pour imprimer des expressions qui ont un délimiteur invisible obtenu par la commande `\right` :

$$y = \begin{cases} +1 & \text{si } d > c, \\ -1 & \text{le matin,} \\ 0 & \text{la journée.} \end{cases}$$

*Default options*

## 1 Общие сведения

Заметьте разницу в стиле верстки выражений в абзацах и выключных:  $\lim_{n \rightarrow \infty} \sum_{k=1}^n \frac{1}{k^2} = \frac{\pi^2}{6} = 1.644934$ :

$$\lim_{n \rightarrow \infty} \sum_{k=1}^n \frac{1}{k^2} = \frac{\pi^2}{6} = 1.644934. \quad (1)$$

**Строчные греческие буквы** вводятся как `\alpha`, `\beta`, `\gamma`, ..., **прописные буквы** вводятся как `\Gamma`, `\Delta`, ...,  $\alpha$ ,  $\beta$ ,  $\gamma$ ,  $\lambda$ ,  $\xi$ ,  $\pi$ ,  $\mu$ ,  $\Phi$ ,  $\Omega$ .

**Оператор интеграла** печатает команда `\int` ( $\int_0^\pi$ ).

$$\int_0^\pi \sqrt{\pi^2 - x^2} dx = \frac{\pi^3}{4} \quad (2)$$

Окружение `array` можно также использовать для верстки выражений, имеющих один большой ограничитель, подставляя `\right` в качестве невидимого правого ограничителя:

$$y = \begin{cases} +1 & \text{если } d > c, \\ -1 & \text{по утрам,} \\ 0 & \text{остальное время дня.} \end{cases}$$

## 2 Généralités

Remarquez le format différent des expressions mathématiques composées « en ligne » ou « hors texte » :  $\lim_{n \rightarrow \infty} \sum_{k=1}^n \frac{1}{k^2} = \frac{\pi^2}{6} = 1.644934$ ,

$$\lim_{n \rightarrow \infty} \sum_{k=1}^n \frac{1}{k^2} = \frac{\pi^2}{6} = 1.644934. \quad (3)$$

Les lettres **grecques minuscules** sont saisies de la manière suivante : `\alpha`, `\beta`, `\gamma`, etc. Les lettres **grecques majuscules** sont saisies ainsi : `\Gamma`, `\Delta`, etc. :  $\alpha$ ,  $\beta$ ,  $\gamma$ ,  $\lambda$ ,  $\xi$ ,  $\pi$ ,  $\mu$ ,  $\Phi$ ,  $\Omega$ .

La commande `\int` produit une **intégrale** ( $\int_0^\pi$ ).

$$\int_0^\pi \sqrt{\pi^2 - x^2} dx = \frac{\pi^3}{4} \quad (4)$$

L'environnement `array` peut également être utilisé pour imprimer des expressions qui ont un délimiteur invisible obtenu par la commande `\right` :

$$y = \begin{cases} +1 & \text{si } d > c, \\ -1 & \text{le matin,} \\ 0 & \text{la journée.} \end{cases}$$

*Options selected: frenchmath, lf, minionint*

Figure 21.22: Multilingual Russian/French math typesetting with Minion Pro

Table 21.11: Font suppliers defined in the *Fontname* scheme.

0 fontfont	FontFont	l linotype	Linotype
2 elsnerflake	Elsner & Flake	m monotype	Monotype
5 softmake	Softmaker	n ibm	IBM
9 unknown		o corel	Corel
a autologi	Autologic	p adobe	Adobe (p for PostScript)
b bitstrea	Bitstream	r -	raw [obsolete]
c cg	Compugraphic	s sun	Storm Type
d dtc	Digital Typeface Corporation	t paragrap	ParaGraph
e apple	Apple	u urw	URW
f public	freely distributable, small foundries	w wolfram	Wolfram
g gnu	Free Software Foundation	z -	bizarre (nonstandard name)
h bh	Bigelow & Holmes	- autofont	Eddie Kohler's autofont
i itc	International Typeface Corporation	- jknappen	Jörg Knappen
j microsoft	Microsoft	- mnm	Hong Feng, free software
k softkey	SoftKey	- yandy	Y&Y

The basic scheme for the filenames is to use eight letters in seven groups

*S TT W [V...] [N] [E] [DD]*

where

- S** is the *supplier* of the font;
- TT** is the *typeface name*;
- W** represents the *weight*;
- V...** represents the *variant(s)*, and is omitted if both it and the width are normal. Many fonts have more than one variant;
- N** is the *encoding*, and is omitted if the encoding is nonstandard. Encodings are described in the section on variants;
- E** represents the width (*expansion*), and is omitted if it is normal;
- DD** is the design size (in decimal), and is omitted if the font is linearly scalable.

The weight, variants, and width are probably all best taken from the original name of the font, rather than trying to relate them to some external standard. Some font families (notably Univers) are constructed with a large number of variants according to rational schemes, but these are the exception rather than the rule.

### 21.7.0.12 Supplier

The “supplier” is the source of a font, typically a (digital) type foundry. You should use the supplier letter for the supplier from which you obtained the font, not the original source, since font resellers typically make modifications to the original design. For example, Avant Garde was designed by Herb Lubalin for ITC, but Adobe also sells it; the name of the font that you get from Adobe should start with p.

A list of known suppliers is given in Table 21.11; the second column shows a suggested abbreviation that can be used to name directories.

### 21.7.0.13 Typeface

In the *Fontname* scheme a “typeface” is a collection of related fonts. Sometimes the same typeface abbreviation is used for fonts with different supplier names if one knows (usually by inspection) that the

Table 21.12: *Fontname* weight codes.

a	Thin Hairline	l	Light
b	Bold	m	Medium
c	Black	p	Poster
d	Demi	r	Regular Roman
h	Heavy Heavyface	s	Semibold
j	ExtraLight	u	Ultra UltraBlack
k	Book	x	ExtraBold ExtraBlack

Table 21.13: *Fontname* width codes.

c	Condensed, Cond	r	Regular, Normal, Medium
e	Expanded	t	Thin
n	Narrow	u	UltraCompressed
o	UltraCondensed	v	ExtraExpanded (between e and w)
p	Compressed, Compact	w	Wide
q	ExtraCompressed, ExtraCondensed	x	Extended, Elongated

fonts truly are the same. See the full documentation for a list of typefaces.

#### 21.7.0.14 Weights

The “weight” of a font specifies its boldness. Arranged from lightest to heaviest we have *hairline*, *extra light*, *light*, *book*, *regular*, *medium*, *demibold*, *semibold*, *bold*, *extra bold*, *heavy*, *black*, *ultra*, and *poster*. An alphabetically ordered list of abbreviations for weights is given in Table 21.12. The names are those typically used in real font names.

#### 21.7.0.15 Widths

The “width” of a font specifies the compression or expansion of the font. Arranged from narrowest to widest we have *ultra compressed*, *extra condensed*, *compressed*, *condensed*, *narrow*, *regular*, *extended*, *expanded*, and *wide*.

Expansion or compression of fonts is sometimes done automatically (as by the PostScript `scale` operator), and sometimes by human designers. In the latter case, the designer also presumably chooses a font name including “Extended” or “Expanded” or “Narrow” or whatever, according to preference, and the abbreviation can follow along. When creating a new synthetically expanded or compressed font for use with T<sub>E</sub>X, e.g., with `afm2tfm` or `fontinst`, use `n` and `e`. Table 21.13 shows the possible widths.

#### 21.7.0.16 Variants

“Variants” include typeface variations (e.g., *italic*, *sans serif*), and font encodings (e.g., Adobe standard, T<sub>E</sub>X text). A fontname may require multiple variants. To resolve the worst ambiguities, any encoding variant (7, 8, or 9, see below) should come last and any other numeral variant come first (to avoid confusion with a design size). It is recommended (though not required) that all other variants be specified in alphabetical order.

- a** Alt, Arrows, Alternative
- c** SmallCaps
- d** Display, Titling, Caption, Headline, TallCaps, SwashCaps, LombardicCaps, Festive
- e** Engraved, Copperplate, Elite



- f** Fraktur, Gothic, OldEnglish, Handtooled
- g** SmallText
- h** Shadow
- i** Italic, Kursiv, Ital
- j** Old-style digits
- k** font-specific Greek encodings only, now obsolete, use “82”, “83”, or “84” instead.
- l** Outline OpenFace Blanks
- m** Math italic (for typeface-specific math encodings only, e.g., *Lucida*, use “7m” instead.)
- n** Informal, Fashion, Schlbk (rarely used)
- o** Oblique, Obl (slanted)
- p** Ornaments
- r** Roman or sans serif. It is included only as a placeholder if no other variants, including encodings, apply, and either the width is not “r” or a design size is present.<sup>1</sup>
- s** Gothic (sans serif)
- t** Monospace (fixed-width typewriter)
- u** Unslanted italic
- v** MathExtension (for typeface-specific math encodings only, e.g., *Lucida*, use “7v” instead.)
- w** Script Handwritten Swash Calligraphy Cursive Tango
- x** built with Adobe expert encoding
- y** MathSymbol (for typeface-specific math encodings only, e.g., *Lucida*, use “7y” instead.)
- z** font-specific Cyrillic encodings only, use “6.” encodings instead.
- 0** inferior
- 1** superior
- 5** *Escape for phonetic encodings*
- 5a** PhoneticAlternate
- 5i** PhoneticIPA
- 5s** sil-IPA
- 5t** TeX-IPA (Fukui Rei,  $\TeX$ ’s T3 encoding)
- 5w** TeXAfricanLatin (wsuipa fonts,  $\TeX$  OT3)
- 5z** User-defined
- 6** *Escape for Cyrillic encodings*
- 6a**  $\TeX$ ’s T2A encoding
- 6b** Cyrillic part of ISO 8859-5, seven bits
- 6c**  $\TeX$ ’s T2C encoding
- 6d** Cyrillic CP866 encoding
- 6i** ISO 8859-5 encoding
- 6k** Cyrillic KOI8-R encoding
- 6m** Cyrillic Macintosh encoding
- 6s** Storm extra encoding
- 6t**  $\TeX$ ’s T2B encoding
- 6w** Cyrillic CP1251 encoding
- 6x**  $\TeX$ ’s X2 encoding
- 6y**  $\TeX$ ’s LCY encoding
- 6z** User-defined

<sup>1</sup>Note that when the normal version of the typeface is sans serif (e.g., Helvetica), “r” should be used when necessary, not “s”, which should only be used when the typeface family has both serif and sans serif variants.

- 7 *Escape for 7-bit encodings*
- 7a A (alternate characters only)
- 7c Dfr (Fraktur)
- 7d OsF, OSF (oldstyle digit encoding)
- 7f Fraction
- 7k OT2Cyrillic
- 7m TeXMathItalicEncoding
- 7t T<sub>E</sub>X text encoding (as in `cmr`)
- 7v TeXMathExtensionEncoding
- 7y TeXMathSymbolEncoding
- 7z User-defined
- 8 *Escape for 8-bit encodings*
- 82 GreekKeys
- 83 Ibycus1
- 84 Ibycus2
- 8a StandardEncoding (Adobe standard encoding)
- 8c TeXTextCompanion (E<sub>T</sub><sub>X</sub>'s TS1 encoding)
- 8e CE (Adobe CE)
- 8f TeXAfricanLatin (E<sub>T</sub><sub>X</sub>'s T4 encoding)
- 8g groff
- 8i TS0 (Intersection of TS1/Adobe Standard)
- 8m Macintosh standard encoding
- 8n LM1 (*Textures*)
- 8q encqxoosix (Q<sub>X</sub>, from GUST)
- 8r TeXBase1Encoding
- 8t CorkEncoding (E<sub>T</sub><sub>X</sub>'s T1, also known as `256.enc`)
- 8u XT2Encoding (`cmtt` and Latin 2)
- 8v TeXVietnamese (being defined)
- 8w *Windows 3.1 ANSI encoding*
- 8x Expert encoding
- 8y LY1 (`texnansi`)
- 8z XL2 encoding (`cmr` and Latin 2)
- 9 *Escape for expert encodings*<sup>1</sup>
- 9c *expert + Text companion*
- 9d *expert + oldstyle digits + Cork*
- 9e Expert + Cork<sup>2</sup>
- 9i TS0X (Intersection of TS1/Standard/Expert)
- 9o Expert, oldstyle digits and T<sub>E</sub>X text
- 9s SuperFont
- 9t Expert and + T<sub>E</sub>X text
- 9u Unicode-compatible
- 9x TeXnANSIEncodingX `texnansx`, `texnansi` without repeats
- 9z User-defined

<sup>1</sup>The “9”escape is often used because the “x” and “j” codes followed by a two-character regular variant makes some names too long, e.g., “9t” is equivalent to “x7t”, and “9d” is equivalent to “jx7t”.

<sup>2</sup>Note that “8x” indicates a font in the “Expert” encoding itself, while “x” signals an *expertised* font, i.e., a composite (virtual) font that includes characters from an “8x” font. In fact an “xee” sequence is replaced by “9e”, to save characters.

- songti (for mnm)

Naming a font with a certain encoding variant does not mean that all the characters appear in precisely the same positions as in the encoding definition. It is sufficient that the usual  $\TeX$  macros can be made to work by remapping or via ligatures.

If a name does not contain a specific encoding variant, its encoding is “unspecified”. For example, some of the fonts distributed with dvips have names like `ptmr` for *Times-Roman*; they use the dvips encoding, which is close to (but not the same as) the  $\TeX$  text encoding (as in *Computer Modern Roman*).

## Examples

As an example of how the Font names are implemented let us look at the directory that contains the `.tfm` files for using various combinations of Adobe’s *Times Roman* font. We first find a bunch of 7-bit encoded files.

```
ptmr7t.tfm  Standard Times-Roman
ptmrc7t.tfm Small Caps Times-Roman
ptmro7t.tfm Oblique Times-Roman
ptmb7t.tfm  Standard Times-Bold
ptmbc7t.tfm Small Caps Times-Bold
ptmbo7t.tfm Oblique Times-Bold
ptmbi7t.tfm Standard Times-BoldItalic
ptmri7t.tfm Standard Times-Italic
```

Similarly, for the eight-bit encodings we have the following (partial) list (We only look at variants of the *Times-Roman* instance.)

```
ptmr8c.tfm  Standard (Text Companion encoding)
ptmr8r.tfm  Standard ( $\TeX$  base1 encoding)
ptmr8rn.tfm Narrow ( $\TeX$  base1 encoding)
ptmr8t.tfm  Standard (Cork or T1 encoding)
ptmr8y.tfm  Standard (LY1 or texnansi encoding, a variant of T1)
ptmrc8t.tfm Small Caps ( $\TeX$  T1 encoding)
ptmrc8y.tfm Small Caps ( $\TeX$  LY1 encoding)
ptmro8c.tfm Oblique (Text Companion encoding)
ptmro8r.tfm Oblique ( $\TeX$  base1 encoding)
ptmro8t.tfm Oblique (T1 encoding)
ptmro8y.tfm Oblique (LY1 encoding)
ptmrr8re.tfm Expanded ( $\TeX$  base1 encoding)
```

In fact, when preparing the PostScript output file, dvips constructs the various widths and slanted variants on the fly by looking in the map file (the following lines are an extract).<sup>1</sup>

```
ptmr8r Times-Roman "TeXBase1Encoding ReEncodeFont" <8r.enc
ptmr8y Times-Roman "TeXnANSIEncoding ReEncodeFont" <texnansi.enc
ptmro8r Times-Roman ".167 SlantFont TeXBase1Encoding ReEncodeFont" <8r.enc
ptmr8rn Times-Roman ".82 ExtendFont TeXBase1Encoding ReEncodeFont" <8r.enc
ptmrr8re Times-Roman "1.2 ExtendFont TeXBase1Encoding ReEncodeFont" <8r.enc
```

The first line (`ptmr8r`) specifies that the font glyphs are encoded according to the  $\TeX$  basic encoding, while the second line (`ptmr8y`) encodes them according to the `texnansi` (LY1) encoding. For the

<sup>1</sup>The keywords `ReEncodeFont`, `SlantFont`, `ExtendFont`, as well as `TeXBase1Encoding` and `TeXnANSIEncoding` correspond to procedure names that are defined in the prolog that dvips prepends to each PostScript file that it generates, see Section 22.2.5 on page 109.

remaining three lines we stay with the T<sub>E</sub>X base1 encoding but ask the PostScript engine to perform a few geometric transformations. First, (`ptmro8r`) we generate a slanted variant (the `.167 SlantFont` command slants the characters 1/6 to the right), then we generate a narrow (`ptmr8rn`, with “`.82 Extendfont`”) and Expanded (`ptmr8re`, with “`1.2 Extendfont`”) variant.

Nowadays, with large OpenType fonts and operating systems no longer limiting the length of file-names, the naming scheme no longer needs to be used so strictly. We have seen this already when discussing the various schemes that have been developed for dealing with OpenType fonts, as described in Sections 21.6.2 and 21.6.1. There the *Fontname* scheme is only maintained for naming the `.fd` files, while font names for the `.tfm` and `.vfm` instances use long explicit names. As an example we show the following (truncated) entries for the file names associated to Cyrillic characters (T<sub>2</sub>a encoding) in the `.map` file of the MinionPro package.

```
MinionPro-Regular-lf-l1-t2a--base MinionPro-Regular ...
MinionPro-Regular-lf-t2a--base MinionPro-Regular ...
MinionPro-Regular-osf-l1-t2a--base MinionPro-Regular ...
MinionPro-Regular-osf-t2a--base MinionPro-Regular " ...
MinionPro-Regular-tlf-l1-t2a--base MinionPro-Regular ...
MinionPro-Regular-tlf-t2a--base MinionPro-Regular ...
MinionPro-Regular-tosf-l1-t2a--base MinionPro-Regular ...
MinionPro-Regular-tosf-t2a--base MinionPro-Regular ...
```

## 21.8 Font encoding tables

Table 21.14: Font encoding table, by name.

Name	Glyph	EC	8r	ISO Latin 1	Adobe Standard	Windows ANSI	Mac	PDF
A	A	65	65	65	65	65	65	65
AE	Æ	198	198	198	225	198	174	198
Aacute	Á	193	193	193		193	231	193
Abreve	Ă	128						
Acircumflex	Â	194	194	194		194	229	194
Adieresis	Ä	196	196	196		196	128	196
Agrave	À	192	192	192		192	203	192
Aogonek	Ą	129						
Aring	Å	197	197	197		197	129	197
Atilde	Ã	195	195	195		195	204	195
B	B	66	66	66	66	66	66	66
C	C	67	67	67	67	67	67	67
Cacute	Ć	130						
Ccaron	Č	131						
Ccedilla	Ç	199	199	199		199	130	199
D	D	68	68	68	68	68	68	68
Dcaron	Ď	132						
E	E	69	69	69	69	69	69	69
Eacute	É	201	201	201		201	131	201
Ecaron	Ě	133						

Name	Glyph	EC	8r	ISO Latin 1	Adobe Standard	Windows ANSI	Mac	PDF
Ecircumflex	Ê	202	202	202		202	230	202
Edieresis	Ë	203	203	203		203	232	203
Egrave	È	200	200	200		200	233	200
Eng	Ě	141						
Eogonek	Ę	134						
Eth	Ð	208	208	208		208		208
F	F	70	70	70	70	70	70	70
G	G	71	71	71	71	71	71	71
Gbreve	Ğ	135						
Germandbls	ß	223						
H	H	72	72	72	72	72	72	72
I	I	73	73	73	73	73	73	73
IJ	IJ	156						
Iacute	Í	205	205	205		205	234	205
Icircumflex	Î	206	206	206		206	235	206
Idieresis	Ï	207	207	207		207	236	207
Idotaccent	İ	157						
Igrave	Ì	204	204	204		204	237	204
J	J	74	74	74	74	74	74	74
K	K	75	75	75	75	75	75	75
L	L	76	76	76	76	76	76	76
Lacute	Ł	136						
Lquoteright	Ł	137						
Lslash	Ł	138	6		232			149
M	M	77	77	77	77	77	77	77
N	N	78	78	78	78	78	78	78
Nacute	Ń	139						
Ncaron	Ň	140						
Ntilde	Ñ	209	209	209		209	132	209
O	O	79	79	79	79	79	79	79
OE	Œ	215	140		234	140	206	150
Oacute	Ó	211	211	211		211	238	211
Ocircumflex	Ô	212	212	212		212	239	212
Odieresis	Ö	214	214	214		214	133	214
Ograve	Ò	210	210	210		210	241	210
Ohungarumlaut	Ő	142						
Oslash	Ø	216	216	216	233	216	175	216
Otilde	Õ	213	213	213		213	205	213
P	P	80	80	80	80	80	80	80
Q	Q	81	81	81	81	81	81	81
R	R	82	82	82	82	82	82	82
Racute	Ŕ	143						
Rcaron	Ř	144						
S	S	83	83	83	83	83	83	83
Sacute	Ś	145						

Name	Glyph	EC	8r	ISO Latin 1	Adobe Standard	Windows ANSI	Mac	PDF
Scaron	Š	146	138			138		151
Scedilla	Œ	147						
T	T	84	84	84	84	84	84	84
Tcaron	Ť	148						
Tcedilla	Ț	149						
Thorn	Þ	222	222	222		222		222
U	U	85	85	85	85	85	85	85
Uacute	Ú	218	218	218		218	242	218
Ucircumflex	Û	219	219	219		219	243	219
Udieresis	Ü	220	220	220		220	134	220
Ugrave	Ù	217	217	217		217	244	217
Uhungarumlaut	Ű	150						
Uring	Û	151						
V	V	86	86	86	86	86	86	86
W	W	87	87	87	87	87	87	87
X	X	88	88	88	88	88	88	88
Y	Y	89	89	89	89	89	89	89
Yacute	Ý	221	221	221		221		221
Ydieresis	ÿ	152	159			159	217	152
Z	Z	90	90	90	90	90	90	90
Zacute	Ż	153						
Zcaron	Ž	154	14					153
Zdotaccent	Ž	155						
a	a	97	97	97	97	97	97	97
aacute	á	225	225	225		225	135	225
abreve	ă	160						
acircumflex	â	226	226	226		226	137	226
acute	´	1	180	180	194	180	171	180
adieresis	ä	228	228	228		228	138	228
ae	æ	230	230	230	241	230	190	230
agrave	à	224	224	224		224	136	224
ampersand	&	38	38	38	38	38	38	38
aogonek	ą	161						
aring	å	229	229	229		229	140	229
asciicircum	^	94	94	94	94	94	94	94
asciitilde	~	126	126	126	126	126	126	126
asterisk	*	42	42	42	42	42	42	42
at	@	64	64	64	64	64	64	64
atilde	ã	227	227	227		227	139	227
b	b	98	98	98	98	98	98	98
backslash	\	92	92	92	92	92	92	92
bar		124	124	124	124	124	124	124
blank							202	
braceleft	{	123	123	123	123	123	123	123
braceright	}	125	125	125	125	125	125	125
bracketleft	[	91	91	91	91	91	91	91

Name	Glyph	EC	8r	ISO Latin 1	Adobe Standard	Windows ANSI	Mac	PDF
bracketright	]	93	93	93	93	93	93	93
breve		8	11	150	198	24	249	24
brokenbar			166	166		166		166
bullet	•		149		183		165	
c	c	99	99	99	99	99	99	99
acute	á	162						
caron	ˇ	7	16	159	207	25	255	25
ccaron	č	163						
cedilla	ç	231	231	231		231	141	231
cedilla	¸	11	184	184	203	184	252	184
cent	¢		162	162	162	162	162	162
circumflex	ˆ	2	136	147	195	136	246	26
colon	:	58	58	58	58	58	58	58
comma	,	44	44	44	44	44	44	44
compoundwordmark		23						
copyright	©		169	169		169	169	169
currency	¤		164	164	168	164	219	164
d	d	100	100	100	100	100	100	100
dagger	†		134		178	134	160	129
daggerdbl	‡		135		179	135	224	130
dbar	đ	158						
degree	°		176	176		176	161	176
diaeresis	¨	4	168	168	200	168	172	168
divide	÷		247	247		247	214	247
dollar	\$	36	36	36	36	36	36	36
dotaccent	˙	10	1	151	199	27	250	27
dotlessi	ı	25	17	144	245		245	154
dotlessj		26	18					
dquoteright	’	164						
e	e	101	101	101	101	101	101	101
acute	é	233	233	233		233	142	233
ecaron	ě	165						
ecircumflex	ê	234	234	234		234	144	234
edieresis	ë	235	235	235		235	145	235
egrave	è	232	232	232		232	143	232
eight	8	56	56	56	56	56	56	56
ellipsis	...		133		188	133	201	131
emdash	—	22	151		208	151	209	132
endash	–	21	150		177	150	208	133
eng	ŋ	173						
eogonek	ė	166						
equal	=	61	61	61	61	61	61	61
eth	ð	240	240	240		240		240
exclam	!	33	33	33	33	33	33	33
exclamdown	¡	189	161	161	161	161	193	161
f	f	102	102	102	102	102	102	102

Name	Glyph	EC	8r	ISO Latin 1	Adobe Standard	Windows ANSI	Mac	PDF
ff	ff	27	19					
ffi	ffi	30	20					
ffl	ffl	31	21					
fi	fi	28	2		174		222	147
five	5	53	53	53	53	53	53	53
fl	fl	29	3		175		223	148
florin	f		131		166	131	196	134
four	4	52	52	52	52	52	52	52
fraction	/		4		164		218	135
g	g	103	103	103	103	103	103	103
gbreve	g̃	167						
germandbls	ß	255	223	223	251	223	167	223
grave	`	0	30	145	193		96	96
greater	>	62	62	62	62	62	62	62
guillemotleft	«	19	171	171	171	171	199	171
guillemotright	»	20	187	187	187	187	200	187
guilsingleft	<	14	139		172	139	220	136
guilsingright	>	15	155		173	155	221	137
h	h	104	104	104	104	104	104	104
hungarumlaut	¨	5	5	157	205	28	253	28
hyphen	-	127	173		45	173	45	45
i	i	105	105	105	105	105	105	105
iacute	í	237	237	237		237	146	237
icircumflex	î	238	238	238		238	148	238
idieresis	ï	239	239	239		239	149	239
igrave	ì	236	236	236		236	147	236
ij	ij	188						
j	j	106	106	106	106	106	106	106
k	k	107	107	107	107	107	107	107
l	l	108	108	108	108	108	108	108
lacute	ĺ	168						
less	<	60	60	60	60	60	60	60
logicalnot	¬		172	172		172	194	172
lquoteright	ŀ	169						
lslash	ł	170	7		248			155
m	m	109	109	109	109	109	109	109
macron	ˉ	9	175	175	197	175	248	175
minus	−		12	45				138
mu	μ		181	181		181	181	181
multiply	×		215	215		215		215
n	n	110	110	110	110	110	110	110
nacute	ń	171						
nbspace				160				
ncaron	ň	172						
nine	9	57	57	57	57	57	57	57
ntilde	ñ	241	241	241		241	150	241



Name	Glyph	EC	8r	ISO Latin 1	Adobe Standard	Windows ANSI	Mac	PDF
numbersign	#	35	35	35	35	35	35	35
o	o	111	111	111	111	111	111	111
oacute	ó	243	243	243		243	151	243
ocircumflex	ô	244	244	244		244	153	244
odieresis	ö	246	246	246		246	154	246
oe	œ	247	156		250	156	207	156
ogonek	˛	12	8	158	206	29	254	29
ograve	ò	242	242	242		242	152	242
ohungarumlaut	ő	174						
one	1	49	49	49	49	49	49	49
onehalf	½		189	189		189		189
onequarter	¼		188	188		188		188
onesuperior	¹		185	185		185		185
ordfeminine	ª		170	170	227	170	187	170
ordmasculine	º		186	186	235	186	188	186
oslash	ø	248	248	248	249	248	191	248
otilde	õ	245	245	245		245	155	245
p	p	112	112	112	112	112	112	112
paragraph	¶		182	182	182	182	166	182
parenleft	(	40	40	40	40	40	40	40
parenright	)	41	41	41	41	41	41	41
percent	%	37	37	37	37	37	37	37
period	.	46	46	46	46	46	46	46
periodcentered	·		183	183	180	183	225	183
perthousand	‰	24	137		189	137	228	139
plus	+	43	43	43	43	43	43	43
plusminus	±		177	177		177	177	177
q	q	113	113	113	113	113	113	113
question	?	63	63	63	63	63	63	63
questiondown	¿	190	191	191	191	191	192	191
quotedbl	"	34	34	34	34	34	34	34
quotedblbase	„	18	132		185	132	227	140
quotedblleft	“	16	147		170	147	210	141
quotedblright	”	17	148		186	148	211	142
quoteleft	‘	96	145	96	96	145	212	143
quoteright	’	39	146	39	39	146	213	144
quotesinglbase	,	13	130		184	130	226	145
quotesingle	'		31		169	39	39	39
r	r	114	114	114	114	114	114	114
racute	í	175						
rcaron	ř	176						
registered	®		174	174		174	168	174
ring	°	6	9	154	202	30	251	30
s	s	115	115	115	115	115	115	115
sacute	ś	177						
scaron	š	178	154			154		157

Name	Glyph	EC	8r	ISO Latin 1	Adobe Standard	Windows ANSI	Mac	PDF
scedilla	§	179						
section	§	159	167	167	167	167	164	167
semicolon	;	59	59	59	59	59	59	59
seven	7	55	55	55	55	55	55	55
sfthyphen				173				
six	6	54	54	54	54	54	54	54
slash	/	47	47	47	47	47	47	47
space			32	32	32	160	32	32
sterling	£	191	163	163	163	163	163	163
t	t	116	116	116	116	116	116	116
tcedilla	ţ	181						
thorn	þ	254	254	254		254		254
three	3	51	51	51	51	51	51	51
threequarters	¾		190	190		190		190
threesuperior	³		179	179		179		179
tilde	~	3	152	148	196	152	247	31
tquoteright	ƒ	180						
trademark	™		153			153	170	146
two	2	50	50	50	50	50	50	50
twosuperior	²		178	178		178		178
u	u	117	117	117	117	117	117	117
uacute	ú	250	250	250		250	156	250
ucircumflex	û	251	251	251		251	158	251
udieresis	ü	252	252	252		252	159	252
ugrave	ù	249	249	249		249	157	249
uhungarumlaut	ű	182						
underscore	—	95	95	95	95	95	95	95
uring	ů	183						
v	v	118	118	118	118	118	118	118
visiblespace		32						
w	w	119	119	119	119	119	119	119
x	x	120	120	120	120	120	120	120
y	y	121	121	121	121	121	121	121
yacute	ý	253	253	253		253		253
ydieresis	ÿ	184	255	255		255	216	255
yen	¥		165	165	165	165	180	165
z	z	122	122	122	122	122	122	122
zacute	ż	185						
zcaron	ž	186	15					158
zdotaccent	ẏ	187						
zero	0	48	48	48	48	48	48	48

Table 21.15: Font encoding table, by number.

Decimal	Octal	Hexadecimal	EC Glyph	EC	TeXBase1	ISO Latin 1	Standard	Windows ANSI	Mac Roman	PDF
0	0	0	`	grave						
1	1	1	´	acute	dotaccent					
2	2	2	^	circumflex	fi					
3	3	3	~	tilde	fl					
4	4	4	¨	dieresis	fraction					
5	5	5	˘	hungarumlaut	hungarumlaut					
6	6	6	°	ring	Lslash					
7	7	7	ˆ	caron	lslash					
8	10	8	˘	breve	ogonek					
9	11	9	ˉ	macron	ring					
10	12	a	˙	dotaccent						
11	13	b	¸	cedilla	breve					
12	14	c	ˆ	ogonek	minus					
13	15	d	‚	quotesingbase						
14	16	e	‹	guilsingleft	Zcaron					
15	17	f	›	guilsingright	zcaron					
16	20	10	“	quotedblleft	caron					
17	21	11	”	quotedblright	dotlessi					
18	22	12	„	quotedblbase	dotlessj					
19	23	13	«	guillemotleft	ff					
20	24	14	»	guillemotright	ffi					
21	25	15	—	endash	ffl					
22	26	16	—	emdash						
23	27	17		compoundwordmark						
24	30	18	◊	perthousand				breve		breve
25	31	19	ı	dotlessi				caron		caron
26	32	1a	j	dotlessj				circumflex		circumflex
27	33	1b	ff	ff				dotaccent		dotaccent
28	34	1c	fi	fi				hungarumlaut		hungarumlaut
29	35	1d	fl	fl				ogonek		ogonek
30	36	1e	ffi	ffi	grave			ring		ring
31	37	1f	ffl	ffl	quotesingle			tilde		tilde
32	40	20	␣	visible space	space	space	space	space	space	space
33	41	21	!	exclam	exclam	exclam	exclam	exclam	exclam	exclam
34	42	22	"	quotedbl	quotedbl	quotedbl	quotedbl	quotedbl	quotedbl	quotedbl
35	43	23	#	numeralsign	numeralsign	numeralsign	numeralsign	numeralsign	numeralsign	numeralsign
36	44	24	\$	dollar	dollar	dollar	dollar	dollar	dollar	dollar
37	45	25	%	percent	percent	percent	percent	percent	percent	percent
38	46	26	&	ampersand	ampersand	ampersand	ampersand	ampersand	ampersand	ampersand
39	47	27	'	quoteright	quoteright	quoteright	quoteright	quotesingle	quotesingle	quotesingle
40	50	28	(	parenleft	parenleft	parenleft	parenleft	parenleft	parenleft	parenleft
41	51	29	)	parenright	parenright	parenright	parenright	parenright	parenright	parenright
42	52	2a	*	asterisk	asterisk	asterisk	asterisk	asterisk	asterisk	asterisk
43	53	2b	+	plus	plus	plus	plus	plus	plus	plus
44	54	2c	,	comma	comma	comma	comma	comma	comma	comma
45	55	2d	-	hyphen	hyphen	minus	hyphen	hyphen	hyphen	hyphen
46	56	2e	.	period	period	period	period	period	period	period
47	57	2f	/	slash	slash	slash	slash	slash	slash	slash
48	60	30	0	zero	zero	zero	zero	zero	zero	zero
49	61	31	1	one	one	one	one	one	one	one
50	62	32	2	two	two	two	two	two	two	two
51	63	33	3	three	three	three	three	three	three	three
52	64	34	4	four	four	four	four	four	four	four
53	65	35	5	five	five	five	five	five	five	five
54	66	36	6	six	six	six	six	six	six	six
55	67	37	7	seven	seven	seven	seven	seven	seven	seven
56	70	38	8	eight	eight	eight	eight	eight	eight	eight

Decimal	Octal	Hexadecimal	EC Glyph	EC	TeXBase1	ISO Latin 1	Standard	Windows ANSI	Mac Roman	PDF
57	71	39	9	nine	nine	nine	nine	nine	nine	nine
58	72	3a	:	colon	colon	colon	colon	colon	colon	colon
59	73	3b	;	semicolon	semicolon	semicolon	semicolon	semicolon	semicolon	semicolon
60	74	3c	<	less	less	less	less	less	less	less
61	75	3d	=	equal	equal	equal	equal	equal	equal	equal
62	76	3e	>	greater	greater	greater	greater	greater	greater	greater
63	77	3f	?	question	question	question	question	question	question	question
64	100	40	@	at	at	at	at	at	at	at
65	101	41	A	A	A	A	A	A	A	A
66	102	42	B	B	B	B	B	B	B	B
67	103	43	C	C	C	C	C	C	C	C
68	104	44	D	D	D	D	D	D	D	D
69	105	45	E	E	E	E	E	E	E	E
70	106	46	F	F	F	F	F	F	F	F
71	107	47	G	G	G	G	G	G	G	G
72	110	48	H	H	H	H	H	H	H	H
73	111	49	I	I	I	I	I	I	I	I
74	112	4a	J	J	J	J	J	J	J	J
75	113	4b	K	K	K	K	K	K	K	K
76	114	4c	L	L	L	L	L	L	L	L
77	115	4d	M	M	M	M	M	M	M	M
78	116	4e	N	N	N	N	N	N	N	N
79	117	4f	O	O	O	O	O	O	O	O
80	120	50	P	P	P	P	P	P	P	P
81	121	51	Q	Q	Q	Q	Q	Q	Q	Q
82	122	52	R	R	R	R	R	R	R	R
83	123	53	S	S	S	S	S	S	S	S
84	124	54	T	T	T	T	T	T	T	T
85	125	55	U	U	U	U	U	U	U	U
86	126	56	V	V	V	V	V	V	V	V
87	127	57	W	W	W	W	W	W	W	W
88	130	58	X	X	X	X	X	X	X	X
89	131	59	Y	Y	Y	Y	Y	Y	Y	Y
90	132	5a	Z	Z	Z	Z	Z	Z	Z	Z
91	133	5b	[	bracketleft	bracketleft	bracketleft	bracketleft	bracketleft	bracketleft	bracketleft
92	134	5c	\	backslash	backslash	backslash	backslash	backslash	backslash	backslash
93	135	5d	]	bracketright	bracketright	bracketright	bracketright	bracketright	bracketright	bracketright
94	136	5e	ˆ	asciicircum	asciicircum	asciicircum	asciicircum	asciicircum	asciicircum	asciicircum
95	137	5f	_	underscore	underscore	underscore	underscore	underscore	underscore	underscore
96	140	60	‘	quoteleft	quoteleft	quoteleft	quoteleft	quoteleft	grave	grave
97	141	61	a	a	a	a	a	a	a	a
98	142	62	b	b	b	b	b	b	b	b
99	143	63	c	c	c	c	c	c	c	c
100	144	64	d	d	d	d	d	d	d	d
101	145	65	e	e	e	e	e	e	e	e
102	146	66	f	f	f	f	f	f	f	f
103	147	67	g	g	g	g	g	g	g	g
104	150	68	h	h	h	h	h	h	h	h
105	151	69	i	i	i	i	i	i	i	i
106	152	6a	j	j	j	j	j	j	j	j
107	153	6b	k	k	k	k	k	k	k	k
108	154	6c	l	l	l	l	l	l	l	l
109	155	6d	m	m	m	m	m	m	m	m
110	156	6e	n	n	n	n	n	n	n	n
111	157	6f	o	o	o	o	o	o	o	o
112	160	70	p	p	p	p	p	p	p	p
113	161	71	q	q	q	q	q	q	q	q
114	162	72	r	r	r	r	r	r	r	r
115	163	73	s	s	s	s	s	s	s	s

Decimal	Octal	Hexadecimal	EC Glyph	EC	TeXBase1	ISO Latin 1	Standard	Windows ANSI	Mac Roman	PDF
116	164	74	t	t	t	t	t	t	t	t
117	165	75	u	u	u	u	u	u	u	u
118	166	76	v	v	v	v	v	v	v	v
119	167	77	w	w	w	w	w	w	w	w
120	170	78	x	x	x	x	x	x	x	x
121	171	79	y	y	y	y	y	y	y	y
122	172	7a	z	z	z	z	z	z	z	z
123	173	7b	{	braceleft	braceleft	braceleft	braceleft	braceleft	braceleft	braceleft
124	174	7c		bar	bar	bar	bar	bar	bar	bar
125	175	7d	}	braceright	braceright	braceright	braceright	braceright	braceright	braceright
126	176	7e	~	asciitilde	asciitilde	asciitilde	asciitilde	asciitilde	asciitilde	asciitilde
127	177	7f	~	hyphen						
128	200	80	Ä	Abreve					Adieresis	
129	201	81	Å	Aogonek					Aring	dagger
130	202	82	Ć	Cacute	quotesinglbase			quotesinglbase	Ccedilla	daggerdbl
131	203	83	Č	Ccaron	florin			florin	Eacute	ellipsis
132	204	84	Ď	Dcaron	quotedblbase			quotedblbase	Ntilde	emdash
133	205	85	Ě	Ecaron	ellipsis			ellipsis	Odieresis	endash
134	206	86	Ę	Eogonek	dagger			dagger	Udieresis	florin
135	207	87	Ğ	Gbreve	daggerdbl			daggerdbl	aacute	fraction
136	210	88	Ĺ	Lacute	circumflex			circumflex	agrave	guilsinglleft
137	211	89	Ł	Lquoteright	perthousand			perthousand	acircumflex	guilsinglright
138	212	8a	Ł	Lslash	Scaron			Scaron	adieresis	minus
139	213	8b	Ń	Nacute	guilsinglleft			guilsinglleft	atilde	perthousand
140	214	8c	Ň	Ncaron	OE			OE	aring	quotedblbase
141	215	8d	Đ	Eng					ccedilla	quotedblleft
142	216	8e	Ö	Ohungarumlaut					eacute	quotedblright
143	217	8f	Ř	Racute					egrave	quoteleft
144	220	90	Ř	Rcaron		dotlessi			ecircumflex	quoteright
145	221	91	Ś	Sacute	quoteleft	grave		quoteleft	edieresis	quotesinglbase
146	222	92	Ŝ	Scaron	quoteright	acute		quoteright	iacute	trademark
147	223	93	Ş	Scedilla	quotedblleft	circumflex		quotedblleft	igrave	fi
148	224	94	Ţ	Tcaron	quotedblright	tilde		quotedblright	icircumflex	fl
149	225	95	Ť	Tcedilla	bullet	macron			idieresis	Lslash
150	226	96	Ű	Uhungarumlaut	endash	breve		endash	ntilde	Oe
151	227	97	Ū	Uring	emdash	dotaccent		emdash	oacute	Scaron
152	230	98	Ÿ	Ydieresis	tilde	dieresis		tilde	ograve	Ydieresis
153	231	99	Ž	Zacute	trademark			trademark	ocircumflex	Zcaron
154	232	9a	Ž	Zcaron	scaron	ring		scaron	odieresis	dotlessi
155	233	9b	Ž	Zdotaccent	guilsinglright	cedilla		guilsinglright	otilde	Lslash
156	234	9c	IJ	IJ	oe			oe	uacute	oe
157	235	9d	İ	Idotaccent		hungarumlaut			ugrave	scaron
158	236	9e	đ	dbar		ogonek			ucircumflex	zcaron
159	237	9f	§	section	Ydieresis	caron		Ydieresis	udieresis	
160	240	a0	ä	abreve		nbspace		space	dagger	
161	241	a1	ą	aogonek	exclamdown	exclamdown	exclamdown	exclamdown	degree	exclamdown
162	242	a2	¢	acutec	cent	cent	cent	cent	cent	cent
163	243	a3	č	ccaron	sterling	sterling	sterling	sterling	sterling	sterling
164	244	a4	ď	dquoteright	currency	currency	fraction	currency	section	currency
165	245	a5	ē	ecaron	yen	yen	yen	yen	bullet	yen
166	246	a6	ę	eogonek	brokenbar	brokenbar	florin	brokenbar	paragraph	brokenbar
167	247	a7	ğ	gbreve	section	section	section	section	germandbls	section
168	250	a8	ĺ	lacute	dieresis	dieresis	currency	dieresis	registered	dieresis
169	251	a9	ł	lquoteright	copyright	copyright	quotesingle	copyright	copyright	copyright
170	252	aa	ł	lslash	ordfeminine	ordfeminine	quotedblleft	ordfeminine	trademark	ordfeminine
171	253	ab	ń	nacute	guillemotleft	guillemotleft	guillemotleft	guillemotleft	acute	guillemotleft
172	254	ac	ň	ncaron	logicalnot	logicalnot	guilsinglleft	logicalnot	dieresis	logicalnot
173	255	ad	ŋ	eng	hyphen	sftthyphen	guilsinglright	hyphen	notequal	

Decimal	Octal	Hexadecimal	EC Glyph	EC	TeXBase1	ISO Latin 1	Standard	Windows ANSI	Mac Roman	PDF
174 256	ae	ø		ohungarumlaut	registered	registered	fi	registered	AE	registered
175 257	af	ř		racute	macron	macron	fl	macron	Oslash	macron
176 260	b0	ŗ		rcaron	degree	degree		degree	infinity	degree
177 261	b1	ŝ		sacute	plusminus	plusminus	endash	plusminus	plusminus	plusminus
178 262	b2	š		scaron	twosuperior	twosuperior	dagger	twosuperior	lessequal	twosuperior
179 263	b3	š		scedilla	threesuperior	threesuperior	daggerdbl	threesuperior	greaterequal	threesuperior
180 264	b4	ť		tquoteright	acute	acute	periodcentered	acute	yen	acute
181 265	b5	ţ		tcedilla	mu	mu		mu	mu	mu
182 266	b6	ű		uhungarumlaut	paragraph	paragraph	paragraph	paragraph	partialdiff	paragraph
183 267	b7	ű		uring	periodcentered	periodcentered	bullet	periodcentered	summation	periodcentered
184 270	b8	ÿ		ydieresis	cedilla	cedilla	quotesinglbase	cedilla	product	cedilla
185 271	b9	ž		zacute	onesuperior	onesuperior	quotedblbase	onesuperior	pi	onesuperior
186 272	ba	ž		zcaron	ordmasculine	ordmasculine	quotedblright	ordmasculine	integral	ordmasculine
187 273	bb	ž		zdotaccent	guillemotright	guillemotright	guillemotright	guillemotright	ordfeminine	guillemotright
188 274	bc	ij		ij	onequarter	onequarter	ellipsis	onequarter	ordmasculine	onequarter
189 275	bd	i		exclamdown	onehalf	onehalf	perthousand	onehalf	Omega	onehalf
190 276	be	ı		questiondown	threequarters	threequarters		threequarters	ae	threequarters
191 277	bf	£		sterling	questiondown	questiondown	questiondown	questiondown	oslash	questiondown
192 300	c0	À		Agrave	Agrave	Agrave		Agrave	questiondown	Agrave
193 301	c1	Á		Aacute	Aacute	Aacute	grave	Aacute	exclamdown	Aacute
194 302	c2	Â		Acircumflex	Acircumflex	Acircumflex	acute	Acircumflex	logicalnot	Acircumflex
195 303	c3	Ã		Atilde	Atilde	Atilde	circumflex	Atilde	radical	Atilde
196 304	c4	Ä		Adieresis	Adieresis	Adieresis	tilde	Adieresis	florin	Adieresis
197 305	c5	Å		Aring	Aring	Aring	macron	Aring	approxequal	Aring
198 306	c6	Æ		AE	AE	AE	breve	AE	Delta	AE
199 307	c7	Ç		Ccedilla	Ccedilla	Ccedilla	dotaccent	Ccedilla	guillemotleft	Ccedilla
200 310	c8	È		Egrave	Egrave	Egrave	dieresis	Egrave	guillemotright	Egrave
201 311	c9	É		Eacute	Eacute	Eacute		Eacute	ellipsis	Eacute
202 312	ca	Ê		Ecircumflex	Ecircumflex	Ecircumflex	ring	Ecircumflex	blank	Ecircumflex
203 313	cb	Ë		Edieresis	Edieresis	Edieresis	cedilla	Edieresis	Agrave	Edieresis
204 314	cc	Ì		Igrave	Igrave	Igrave		Igrave	Atilde	Igrave
205 315	cd	Í		Iacute	Iacute	Iacute	hungarumlaut	Iacute	Otilde	Iacute
206 316	ce	Î		Icircumflex	Icircumflex	Icircumflex	ogonek	Icircumflex	OE	Icircumflex
207 317	cf	Ï		Idieresis	Idieresis	Idieresis	caron	Idieresis	oe	Idieresis
208 320	d0	Ð		Eth	Eth	Eth	emdash	Eth	endash	Eth
209 321	d1	Ñ		Ntilde	Ntilde	Ntilde		Ntilde	emdash	Ntilde
210 322	d2	Ò		Ograve	Ograve	Ograve		Ograve	quotedblleft	Ograve
211 323	d3	Ó		Oacute	Oacute	Oacute		Oacute	quotedblright	Oacute
212 324	d4	Ô		Ocircumflex	Ocircumflex	Ocircumflex		Ocircumflex	quoteleft	Ocircumflex
213 325	d5	Õ		Otilde	Otilde	Otilde		Otilde	quoteright	Otilde
214 326	d6	Ö		Odieresis	Odieresis	Odieresis		Odieresis	divide	Odieresis
215 327	d7	Œ		OE	multiply	multiply		multiply	lozenge	multiply
216 330	d8	Ø		Oslash	Oslash	Oslash		Oslash	ydieresis	Oslash
217 331	d9	Ù		Ugrave	Ugrave	Ugrave		Ugrave	Ydieresis	Ugrave
218 332	da	Ú		Uacute	Uacute	Uacute		Uacute	fraction	Uacute
219 333	db	Û		Ucircumflex	Ucircumflex	Ucircumflex		Ucircumflex	currency	Ucircumflex
220 334	dc	Ü		Udieresis	Udieresis	Udieresis		Udieresis	guilsinglleft	Udieresis
221 335	dd	Ý		Yacute	Yacute	Yacute		Yacute	guilsinglright	Yacute
222 336	de	Þ		Thorn	Thorn	Thorn		Thorn	fi	Thorn
223 337	df	Š		Germandbls	germandbls	germandbls		germandbls	fl	germandbls
224 340	e0	à		agrave	agrave	agrave		agrave	daggerdbl	agrave
225 341	e1	á		aacute	aacute	aacute	AE	aacute	periodcentered	aacute
226 342	e2	â		acircumflex	acircumflex	acircumflex		acircumflex	quotesinglbase	acircumflex
227 343	e3	ã		atilde	atilde	atilde	ordfeminine	atilde	quotedblbase	atilde
228 344	e4	ä		adieresis	adieresis	adieresis		adieresis	perthousand	adieresis
229 345	e5	å		aring	aring	aring		aring	Acircumflex	aring
230 346	e6	æ		ae	ae	ae		ae	Ecircumflex	ae
231 347	e7	ç		ccedilla	ccedilla	ccedilla		ccedilla	Aacute	ccedilla

Decimal	Octal	Hexadecimal	EC Glyph	EC	TeXBase1	ISO Latin 1	Standard	Windows ANSI	Mac Roman	PDF
232 350	e8	è		egrave	egrave	egrave	Lslash	egrave	Edieresis	egrave
233 351	e9	é		eacute	eacute	eacute	Oslash	eacute	Egrave	eacute
234 352	ea	ê		ecircumflex	ecircumflex	ecircumflex	OE	ecircumflex	Iacute	ecircumflex
235 353	eb	ë		edieresis	edieresis	edieresis	ordmasculine	edieresis	Icircumflex	edieresis
236 354	ec	ì		igrave	igrave	igrave		igrave	Idieresis	igrave
237 355	ed	í		iacute	iacute	iacute		iacute	Igrave	iacute
238 356	ee	î		icircumflex	icircumflex	icircumflex		icircumflex	Oacute	icircumflex
239 357	ef	ï		idieresis	idieresis	idieresis		idieresis	Ocircumflex	idieresis
240 360	f0	ð		eth	eth	eth		eth	apple	eth
241 361	f1	ñ		ntilde	ntilde	ntilde	ae	ntilde	Ograve	ntilde
242 362	f2	ò		ograve	ograve	ograve		ograve	Uacute	ograve
243 363	f3	ó		oacute	oacute	oacute		oacute	Ucircumflex	oacute
244 364	f4	ô		ocircumflex	ocircumflex	ocircumflex		ocircumflex	Ugrave	ocircumflex
245 365	f5	õ		otilde	otilde	otilde	dotlessi	otilde	dotlessi	otilde
246 366	f6	ö		odieresis	odieresis	odieresis		odieresis	circumflex	odieresis
247 367	f7	œ		oe	divide	divide		divide	tilde	divide
248 370	f8	ø		oslash	oslash	oslash	lslash	oslash	macron	oslash
249 371	f9	ù		ugrave	ugrave	ugrave	oslash	ugrave	breve	ugrave
250 372	fa	ú		uacute	uacute	uacute	oe	uacute	dotaccent	uacute
251 373	fb	û		ucircumflex	ucircumflex	ucircumflex	germandbls	ucircumflex	ring	ucircumflex
252 374	fc	ü		udieresis	udieresis	udieresis		udieresis	cedilla	udieresis
253 375	fd	ý		yacute	yacute	yacute		yacute	hungarumlaut	yacute
254 376	fe	þ		thorn	thorn	thorn		thorn	ogonek	thorn
255 377	ff	ß		germandbls	ydieresis	ydieresis		ydieresis	caron	ydieresis





# PostScript and PDF tools

Much of this book deals with using PostScript (or PDF) in some way, whether in setting fonts, drawing pictures, or creating color. In this chapter we look at the higher-level relationship and discuss how to generate, manipulate, view and transform PostScript and PDF files. Among the enormous number of packages available for these purposes, we concentrate primarily on open source tools that are likely to be available to most  $\text{\LaTeX}$  users.

The chapter starts with a comparison of the PostScript, PDF, and SVG languages (Section 22.1). We then describe `dvips`, a DVI-to-PostScript translator and its syntax (Section 22.2), `ghostscript`, a PostScript and PDF interpreter and its associated viewers (Section 22.3), and some interesting PostScript manipulation tools (Section 22.4). Finally, we discuss several ways to generate PDF from  $\text{\LaTeX}$  (Section 22.5) and describe a few interesting PDF manipulation tools (Section 22.6).

## 22.1 Display languages: PostScript, PDF, and SVG

After typesetting an electronic document, one usually would like to view the generated output “page”—on paper via a printing device, on a PC screen, with a dedicated program or inside a browser, or (why not?) on your personal digital assistant (PDA) or your portable phone.

Several display languages have been developed over the years. For printing devices PostScript, which is essentially a language for describing a static output page, has become the most important player. In the early 1990s, Adobe developed a light-weight version of PostScript, called the Portable Document Format (PDF) [3]. PDF implements a similar imaging model as PostScript but introduces a more structured format to improve performance for interactive viewing and for adding annotations incrementally. It also adds links and annotations for navigation.

The increasing affordability of the personal computer has drastically reduced the production cost of electronic documents. The World Wide Web makes distributing these documents worldwide cheap, easy, and fast. The development of the XML family of standards has made it possible to apply a unified approach to handle the huge amount of information stored electronically and to transform it into various customizable presentation forms.

Various techniques are now available to transform  $\text{\LaTeX}$  documents into PDF, HTML (XHTML), or XML so that the information can be made available on the web (several chapters of *LaTeX Web Companion* [6] are dedicated to explaining such techniques). A particularly interesting approach, men-

tioned below, involves transforming  $\text{\LaTeX}$ -encoded information into a Scalable Vector Graphics (SVG) format.

This section gives a short introduction to the PostScript, PDF, and SVG languages. We also show the typeset result of a small  $\text{\LaTeX}$  document expressed in these three languages.

## 22.1.1 The PostScript language

### 22.1.1.1 A short history

The history of PostScript starts at Xerox Parc, the research institute of Xerox, where many of the computer technologies that are now widespread were originally developed.<sup>1</sup> In 1980 collaborators of Xerox completed the Interpress page description language. It allowed workstations to communicate with multiple printers. John Warnock and Charles Geschke, two engineers working in the Interpress team tried to convince Xerox to commercialize the language, but after two years of no success they left Xerox and created Adobe.

At first, Adobe wanted to build their own powerful printer but after some thought efforts were redirected towards the development of tools that would control printers manufactured by other companies. The result was the PostScript language, whose first version was released in 1984.

#### • 1984: PostScript level 1

PostScript is a powerful stack-based computer language, that needs a quite powerful system to run on. In fact, during the middle 1980s PostScript printers were often more powerful than the (Macintosh) computers they served.

The main differences between PostScript and its competitors were its *device independence* (it runs on any PostScript device, from the lowest to the highest resolution, and thus frees users from being tied to a given printer manufacturer). From the beginning, Adobe published the syntax of the PostScript language (hence everybody could write software to interpret PostScript code) and licensed its PostScript “RIP”<sup>2</sup> to any manufacturer interested in it.

But the real killer was the combination of Apple’s LaserWriter printer, PostScript (the printer’s controlling language) that could produce typesetter quality output, and Aldus’ PageMaker, a page layout program that allowed authors to take full advantage of the Mac graphical interface. Their coming together signaled the start of the desktop publishing era.

The potential of PostScript was immediately recognized by other manufacturers and PostScript was soon implemented in most printing devices, thus turning PostScript into the global common output language of graphics and prepress programs.

#### • 1991: PostScript level 2

Adobe released a rather significant upgrade of PostScript (level 2) around 1991. Its most important features were improved speed and reliability (better memory management and optimization of the

<sup>1</sup>A short history is at <http://www.parc.xerox.com/about/history/default.html>. Technologies first introduced at Xerox Parc include laser printers, the graphical user interface (icons, pop-up windows, WYSIWYG technology) leading to the personal workstation, Smalltalk, the first object-oriented programming language with an integrated user interface, overlapping windows, integrated documents, a cut & paste editor, and ethernet, which today has become the global standard for interconnecting computers on local-area networks.

<sup>2</sup>A RIP (Raster Image Processor) is a program which translates a PostScript file into a high-resolution raster image, composed of individual dots that the imaging device (a printer, imagesetter, or computer screen) can output. RIPs come in firmware, hardware, or software versions. A firmware RIP is built into a device, e.g., a desktop printer. The hardware RIP is a dedicated piece of hardware configured to process digital files, e.g., in an imagesetter. The software RIP is an independent program that works on any system where it can be compiled, e.g., the publicly available Ghostscript utility does a good job at interpreting PostScript.

interpreter code), on-RIP color separation (for speed and portability), on-RIP image compression (e.g., JPEG), composite fonts (important for Asian languages with large character sets), better support for output devices (large set of printer description files) and improved screening algorithms.

### •1999: PostScript level 3

PostScript level 3 was published in February 1999. The main additions are support for more than 256 graylevels per color; further advances in on-RIP color separation (introduction of new color spaces, in-RIP trapping), more halftones, smooth shading, and support for PDF (PostScript 3 RIPs can interpret PDF files natively).

#### 22.1.1.2 PostScript: an overview of its feature

PostScript [2] ([www.adobe.com/products/postscript/pdfs/PLRM.pdf](http://www.adobe.com/products/postscript/pdfs/PLRM.pdf)) is a page description language. It provides a method for expressing the appearance of a printed page, including text, lines, and graphics.

A device- and resolution-independent, general-purpose, programming language, PostScript describes a complete “output page”. The language is stack oriented and uses “reverse Polish” or postfix notation. It includes looping constructs, procedures, and comparison operators, and it supports many data types, including reals, Booleans, arrays, strings, and complex objects such as dictionaries.

PostScript programs are generally written in the form of ASCII source text, which is easy to create, edit, transmit, and manipulate. Because PostScript is resolution and device independent, the same ASCII file can be viewed on a computer display with a previewer, such as ghostview, and printed on a small laser printer or a high-resolution phototypesetter.

In the PostScript language the following features can be freely combined:

- Arbitrary shapes, which can be constructed from lines, arcs, and cubic curves. The shapes may self-intersect and contain disconnected sections and holes.
- Painting primitives, which permit shapes to be outlined with lines of any thickness, filled with any color, or used as a clipping path to crop any other graphic.
- Text characters, which are treated as graphical shapes that may be operated on by any of the language’s graphics operators. This is fully true for PostScript Type 3 fonts, where character shapes are defined as ordinary PostScript language procedures. In contrast, Adobe’s PostScript Type 1 format defines a special smaller language where character shapes are defined by using specially encoded procedures<sup>1</sup>
- Images (such as photographs or synthetically generated images), which can be sampled at any resolution and with a variety of dynamic ranges. PostScript provides facilities to control the rendering of images on the output device.
- Several color models (device based: RGB, HSB, CMYK; standard based: CIE) and conversion functions from one model to another.
- A general coordinate system facility, which supports all combinations of linear transformations, including scaling, rotation, reflection, and skewing. These transformations apply uniformly to all page elements, including text, graphical images, and sampled images.
- Dictionaries for color spaces, fonts, forms, images, half-tones, and patterns.
- Compression filters, such as JPEG and LZW.

<sup>1</sup>For complex languages with many thousands of characters (e.g., Chinese and Japanese), composite Type 0 fonts can be used. See Sections 21.1.1, 21.1.4, and 21.2.6 for more information on PostScript font technology.

### 22.1.2 PDF: the Portable Document Format

Adobe's Portable Document Format (PDF) [3]<sup>1</sup> is a direct descendant of the PostScript language. Whereas PostScript is a full-blown programming language, PDF is a second-generation, more lightweight graphics language optimized for faster download and display. Most of the advantages of PostScript remain: PDF guarantees page fidelity, down to the smallest glyph or piece of white space, while being portable across different computer platforms. For these reasons, PDF is being used ever more frequently in the professional printing world as a replacement for PostScript. Moreover, all present-day browsers will embed or display PDF material, alongside HTML, using plug-in technology.

The main differences between PostScript and PDF are the following:

- PDF has no built-in programming language functions, i.e., in general PDF cannot calculate values.
- PDF guarantees full page independence by clearly separating resources from page objects.
- PDF files are compact and fully searchable.
- Interactive hyperlinks make PDF files easy to navigate.
- PDF's security features allow PDF documents to have special access rights and digital signatures applied.
- Font outlines need not be included in the file, because PDF files carry sufficient font information to allow PDF-enabled applications (e.g., Adobe's Acrobat Reader) to mimic the appearance of a font.
- PDF has advanced compression features to keep the size of PDF files small and PNG and JPEG images can be inserted directly.
- PDF 1.4, released in November 2001, and later versions support a transparent imaging model (PostScript uses an opaque model) and feature multimedia support. They also introduce tagged PDF, a stylized form of PDF that contains information on content and structure. Tagged PDF lets applications extract and reuse page data (text, graphics, images). For instance, tagged PDF allows text to reflow for display on handheld devices, such as Palm OS or Pocket PC systems.
- PDF 1.5, released in August 2003, includes features for further optimizing multimedia delivery.
- PDF 1.6, released in November 2004, adds enhancements in the field of encryption, and has further improved support in the areas of, amongst others, color spaces, embedding OpenType fonts, markup annotations, and digital signatures.
- PDF 1.7, released in November 2006, introduces new features to increase the control the PDF viewing application has over the appearance and behavior of 3D artwork. It also includes additions to markup annotations to aid technical communication and review, additions to Tagged-PDF to identify the roles of more types of page content, and additions to document navigation to simplify specifying the viewing and organizational characteristics of portable collections, that are used to present, sort, and search collections of related documents, such as email archives, photo collections, etc. PDF 1.7 also improves author's control on digital signatures and on requirements PDF consumer applications must satisfy. Finally, PDF 1.7 guarantees more cross-platform and cross-application stability, by providing encoding information for strings and file names.
- On January 29, 2007, Adobe decided to submit the PDF 1.7 specification to the International Standardization Organization (ISO) as standard document *ISO 32000*. This document contains a reformatted version of the Adobe PDF 1.7 Reference Guide, which also guarantees that the content is vendor neutral, more precise and conforming to ISO conventions.

PDF can be viewed and printed on many different computer platforms by downloading and

<sup>1</sup>The various versions of the *PDF Reference Guide*, including the latest, are available from Adobe's PDF Technology Center website [http://www.adobe.com/devnet/pdf/pdf\\_reference.html](http://www.adobe.com/devnet/pdf/pdf_reference.html).

installing Adobe's Acrobat Reader.<sup>1</sup> Other PDF viewers exist as well. The best-known free ones are ghostscript (see Section 22.3), Evince (see Section 22.3.4), and xpdf.<sup>2</sup>

### 22.1.3 SVG for Scalable Vector Graphics

As the web has grown in popularity and complexity, users and content providers have sought ever better, more precise, and, above all, scalable graphical rendering. As a complement to PDF, the World Wide Web Consortium has developed SVG,<sup>3</sup> an open-standard vector graphics language for describing two-dimensional graphics using XML syntax. It lets you produce web pages containing high-resolution computer graphics.

As an XML instance, SVG consists of Unicode text. It features the usual vector graphics functions. Its fundamental primitive is the *graphics object*, whose model contains the following:

- Graphics paths consisting of polylines, Bézier curves, and other elements:
  - simple or compound, closed or open;
  - (gradient) filled, (gradient) stroked;
  - can be used for clipping;
  - can be used for building common geometric shapes.
- Patterns and markers.
- Templates and symbol libraries.
- Transformations:
  - default coordinate system:  $x$  is right,  $y$  is down,<sup>4</sup> the unit is one pixel;
  - a  $2 \times 3$  transformation matrix lets you specify how you want to translate, rotate, scale or skew the coordinate system;
  - can be nested;
  - viewport maps an area in world coordinates to an area on screen.
- Inclusion of bitmap or raster images.
- Clipping, filter, and raster effects; alpha masks.
- Animations, scripts, and extensions.
- Groupings and styles.
- SVG fonts (independent from fonts installed on the system).

Several DVI drivers exist to translate a DVI file into SVG:

- **dvi2svg** (<http://www.activemath.org/~adrianf/dvi2svg>) developed by Adrian Frischauf;
- **dvisvg** (<http://dvisvg.sourceforge.net>) developed by Rudolf Sabo;
- **dvisvgm** (<http://dvisvgm.sourceforge.net>) developed by Martin Giesekeing.

<sup>1</sup>Freely downloadable from <http://www.adobe.com/products/reader/>.


<sup>2</sup>See <http://www.foolabs.com/xpdf/home.html>.

<sup>3</sup>SVG stands for *Scalable Vector Graphics*. The W3C web site (<http://www.w3.org/Graphics/SVG>) is a good first source of information on SVG and has a lot of pointers to other sites. The current specification (version 1.1) of the SVG language is available at <http://www.w3.org/TR/SVG11/>.

<sup>4</sup>The reference point of the display area in SVG is the upper-left corner. For PostScript, where  $y$  runs upward, the reference point of the page is the lower-left corner.

### 22.1.4 Comparing an example of PostScript, PDF, and SVG

To see in which way PostScript, PDF, and SVG differ let us consider the following small  $\text{\TeX}$  example that typesets a short sentence in Helvetica inside a red rectangle with lines 1 mm wide and filled with a yellow background.



Good typography is fun!

```
\usepackage{xcolor}
\renewcommand{\rmdefault}{phv}
\pagestyle{empty}
\setlength{\fboxrule}{1mm}
\setlength{\fboxsep}{1mm}
\fcolorbox{red}{yellow}{Good typography is fun!}
```

Exa.  
22-1-1

#### 22.1.4.1 The PostScript instance

The above example is equivalent to the following PostScript code.

```
1  %!PS-Adobe-3.0 EPSF-3.0
2  %%BoundingBox: 25 25 142 46
3  %    mystring contains text string to be output
4  /mystring (Good typography is fun!) def
5  /mm {2.83 mul} def % 1 mm = 2.83 points
6  /fontsize 10 def % the size of the font is 10 pt
7  /box {newpath % stack: llx lly
8      moveto
9      wx 0 rlineto
10     0 wy rlineto
11     wx neg 0 rlineto
12     closepath} def
13 /x0 10 mm def
14 /y0 10 mm def
15 /Helvetica findfont fontsize scalefont setfont
16 /wx mystring stringwidth pop 2 mm add def
17 /wy fontsize 2 mm add def
18 gsave % Paint background box in yellow
19   0 0 1 0 setcmykcolor
20   x0 y0 box
21   fill
22 grestore
23 gsave % Draw 1 mm tick red box around it
24   1 mm setlinewidth
25   1 0 0 setrgbcolor
26   x0 y0 box stroke
27 grestore
28 gsave % Get depth of characters below baseline
29   newpath
30   0 0 moveto
31   (y) true charpath flattenpath pathbbox
32   pop pop /lly exch def pop
33 grestore
34 % Go to start point of box and write text
35 x0 y0 moveto
36 % Move 1 mm right and 1 mm plus depth up
37 1 mm 1 mm lly neg add rmoveto mystring show
```

In PostScript lines starting with a percent sign (%) are comment lines and are ignored by a PostScript interpreter. However, the two first lines are special comments: a declaration that we are dealing with a PostScript level 3 file that is encapsulated (line 1), and a specification of the dimensions, in PostScript points, of the bounding box of the image (line 2). Such comments are mainly useful for external programs, e.g.,  $\text{\TeX}$ , which have to know the space to leave on the output page to place the included material (in fact, that is how  $\text{\TeX}$  reserves the space for including the result of typesetting Example 22-1-1, and all other examples in this book).

Then the actual PostScript code starts with a definition of the variable `mystring` that contains the text to be printed “Good typography is fun!” (line 4, line 3 is a user comment and is ignored). We then define the command `mm` which defines that to express millimeters in PostScript points (the PostScript default unit) one has to use the multiplication factor 2.83 (line 5), and we set the `fontsize` variable to 10 points (line 6). The `box` command (lines 7–12) draws a rectangle of width `wx` and height `wy`. It has to be called with the `x` and `y` coordinates of the lower left corner of the rectangle on the stack.

After these preliminary definitions we define (lines 13–14) the coordinates of the lower left corner of the rectangle (`x0=10 mm`, `y0=10 mm`), and set the font to Helvetica (line 15) at a size of 10 pt (using `fontsize` defined on line 6). Now that the font is set we can calculate the width of the text to be typeset (line 16, which uses the string `mystring`, defined on line 4), and set the height to the font size plus 2 mm (line 17), since we want a separation of 1 mm between the text and each side of the rectangle.

The output is prepared in three stages: Firstly, we draw the yellow background of the rectangle (lines 18–22). We use the CMYK model and activate only the third (i.e., pure yellow) component (line 19) before drawing in memory the rectangle with the `box` command which consumes the coordinates (`x0,y0`) which were put on the stack (line 20), and then fill it with the (yellow) color (line 21). Secondly, we draw the sides of the rectangle (line 23–27). After setting the line width to 1 mm (line 24) and the color using the RGB model to “red” (line 25) the rectangle is once again drawn in memory and its sides are stroked (line 26). Thirdly, we want to draw the text inside the rectangle. But first, in order to position the text correctly vertically inside the rectangle we must take into account the extend of the descenders below the baseline of the text (lines 28–33). Therefore we choose the character in the text string which has the lowest descender (“y” in this case) and calculate its bounding box (line 31). We get hold of the lower `y`-coordinate of the character and save it in the variable `lly` (line 32). Finally, we are ready (line 35) to write the text inside the rectangle at the coordinate (`x0, y0`). To position the text string we move right 1 mm and up by 1 mm plus the (negative of the) depth of the descender and show the string (line 37).

#### 22.1.4.2 The PDF instance

We have translated the previous PostScript code into PDF with `epstopdf` (see page 134), and transformed it into a readable format with the `pdftk` utility and its `uncompress` function (see Section 22.6.1). We obtain the following output.

```

1  %PDF-1.3
2  %ääïó
3  1 0 obj
4  <<
5  /Pages 2 0 R
6  /Type /Catalog
7  >>
8  endobj
9  2 0 obj
10 <<
11 /Kids [3 0 R]
12 /Count 1
13 /Type /Pages
14 >>
15 endobj
16 3 0 obj
17 <<
18 /Parent 2 0 R
19 /Resources
20 <<
21 /ExtGState 4 0 R
22 /Font 5 0 R
23 /ProcSet [/PDF /Text]
24 >>
25 /MediaBox [0 0 117 21]
26 /pdftk_PageNum 1
27 /Contents 6 0 R

```

```

28  /Type /Page
29  >>
30  endobj
31  4 0 obj
32  <<
33  /R7 7 0 R
34  >>
35  endobj
36  5 0 obj
37  <<
38  /R8 8 0 R
39  >>
40  endobj
41  6 0 obj
42  <<
43  /Length 233
44  >>
45  stream
46  q 0.1 0 0 0.1 0 0 cm
47  /R7 gs
48  0 0 1 0 k
49  1 i
50  32.9961 32.9961 1118.25 156.598 re
51  f
52  28.3 w
53  1 0 0 RG
54  32.9961 32.9961 1118.25 156.598 re
55  S
56  0 G
57  0 g
58  q
59  10 0 0 10 0 0 cm BT
60  /R8 10 Tf
61  1 0 0 1 6.1293 8.30937 Tm
62  (Good typography is fun!)Tj
63  ET
64  Q
65  Q
66
67  endstream
68  endobj
69  7 0 obj
70  <<
71  /Type /ExtGState
72  /OPM 1
73  >>
74  endobj
75  8 0 obj
76  <<
77  /BaseFont /Helvetica
78  /Subtype /Type1
79  /Type /Font
80  >>
81  endobj
82  9 0 obj
83  <<
84  /Producer (ESP Ghostscript 815.04)
85  /ModDate (D:20071104141918)
86  /CreationDate (D:20071104141918)
87  >>
88  endobj
89  xref
90  0 10
91  0000000000 65535 f
92  0000000015 00000 n
93  0000000066 00000 n
94  0000000125 00000 n
95  0000000705 00000 n
96  0000000738 00000 n
97  0000000299 00000 n
98  0000000586 00000 n
99  0000000633 00000 n
100 0000000771 00000 n
101 trailer

```



```

102
103 <<
104 /Info 9 0 R
105 /Root 1 0 R
106 /Size 10
107 /ID [(...) (...)]
108 >>
109 startxref
110 889
111 %%EOF

```

A canonical PDF file contains at least four parts:

- A *one-line header*, which identifies the PDF specification version used to build the file (line 1).<sup>1</sup>
- A *body* that contains all the objects that make up the document (lines 3–88).
- A *cross-reference table* which describes all the indirect objects in the file (lines 89–100).
- A *trailer* with the location of the cross-reference table and of certain special objects in the body of the file (lines 101–110).

The *body* of the file consists of a series of *objects* (delimited by `obj...endobj` keywords). Objects can contain many types of data. In our case we use only *dictionary* objects (between `<<...>>` delimiters) and *stream* objects (delimited by `stream...endstream` keywords, see lines 45–67), and characterized by a length (line 43). All objects in our example are *indirect objects*, which have a unique *object identifier*, by which other objects can refer to it. This identifier consists of two parts: a positive *object number* and a non-negative *generation number*. In our example we have eight objects, all with generation number zero (e.g., object “1” starts on line 3, object “2” on line 9, etc.).

Object “1” (lines 3–8) is the *root* of the PDF document structure (it is referenced in the *trailer* on line 105). It is of type `/Catalog` (line 6) and points to the root of the *page tree* (the `/Pages` entry in the dictionary on line 5, which refers to object “2”).

Object “2” (lines 9–15), of type `/Pages` (line 13), has only one descendant leaf node (`/count` entry has the value 1, see line 12) and hence one immediate child node (`/Kids` entry, referring to object “3”, see line 11).

Object “3” (lines 16–30) is of type `/Page` (line 28), its parent is object “2” (line 18), and its boundaries in user space units (similar to the bounding box in EPS files) are given by the `/MediaBox` entry (line 25). Furthermore, the page needs the resources enumerated on lines 21–23: `/ExtGState`, describing the graphics state (refers to object “4”), `/Font`, describing the fonts (refers to object “5”), and `/ProcSet`, defining the procedure set names used in the content streams of the page (`/PDF` for painting and graphics state and `/Text` for text procedures). Finally, the `/Contents` entry (line 27) describes the contents of the page (refers to object “6”).

Object “4” (lines 31–35) defines the variable `/R7` as a reference to object “7”.

Object “5” (lines 36–40) defines the variable `/R8` as a reference to object “8”.

Object “6” (lines 41–68) describes the page contents (as announced on line 27). It is a stream object (line 45), whose length is specified as the `/Length` entry in the dictionary (lines 42–44) to be 233 bytes. The remaining lines of the object draw the various items and write the text. The graphics state is saved before defining the transformation matrix, which introduces a scaling factor of 10 between the coordinates used in the file and the user space (line 46), and setting the graphics state parameters (reference to object “7”, see line 47). The color is set to yellow (using the CMYK model, line 48), the flatness tolerance to “1” (line 49) and a rectangle is drawn in memory (line 50) and filled (line 51). The line width is set to 1 mm (line 52) and the color to red (using the RGB model, line 53) before drawing the same rectangle (line 54) but now stroking its sides (line 55). Then, the color for stroking and non-stroking operations is set to black (lines 56–57) and the graphics state is saved once more (line 58). The

<sup>1</sup>The second comment on line 2 contains four binary characters to help ensure proper behaviour of file transfer applications[3, Section 3.4.1 *File Header*].

transformation matrix is redefined and the start of a text object is initiated (line 59). After defining the font by a reference to object “8” (line 60) the text matrix is set (line 61) and the text drawn (line 62). The text object (declared on line 59) is ended (line 63), and the graphics states are restored (line 64 restores the graphics state saved on line 58, and line 65 the one saved on line 46).

Object “7” (lines 69–74) set the graphics state for drawing in “overpaint mode” (line 72). This was used for filling and painting the rectangle, so that the red sides are painted without transparency on top of the yellow background (line 55).

Object “8” (line 75–81) defines a font dictionary of subtype `/Type1` (line 78). The `/BaseFont` entry specifies the PostScript name of the font, in this case *Helvetica* (line 77), which is one of the 14 *standard fonts*, and hence can be used without specifying further information.

Object “9” (lines 82–88) is an informational record about the PDF file, specifying its producer (line 84), and its creation (line 86) and modification (line 85) dates.

The remaining part of the file contains the *cross-reference table* (lines 89–100). It starts with a static entry (line 91) that is followed by nine entries (lines 92–100) which specify the start byte of objects “1” to “9” in the file. Next comes the *trailer* (lines 101–108) with references to the `/Info` object “9” (line 104), the `/Root` object “1” (line 105), the size of the cross-reference table (line 106), and a constructed unique identifier `/ID` (not shown on line 107).<sup>1</sup> Finally, the byte offset from the beginning of the file to the beginning of the cross-reference section (line 89) is given by the number following the `startxref` keyword (line 110).

### 22.1.4.3 SVG instance

XML is a declarative metalanguage and SVG is an instance of an XML grammar, which implements a vocabulary optimized for the description of vector graphics.

```

1  <?xml version="1.0"?>
2  <!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
3    "http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">
4  <svg width="142pt" height="46pt"
5    xmlns="http://www.w3.org/2000/svg" version="1.1">
6    <!-- Draw rectangle sides in red and fill with yellow -->
7    <rect x="25pt" y="25pt" width="113pt" height="19pt"
8      fill="yellow" stroke="red" stroke-width="1mm" />
9    <!-- Write the text in the rectangle -->
10   <text x="30pt" y="37.5pt" font-family="Helvetica"
11     font-size="10pt" fill="black">
12     Good typography is fun!
13   </text>
14 </svg>

```

After the usual XML declaration (line 1) and the definition of its grammar by specifying its DTD (lines 2–3), we encounter the `svg` element, which brackets the whole file (lines 4–14). The start tag of the `svg` element (lines 4–5) defines the dimensions and viewport of the graphics image, the namespace, and the version of SVG to which the element vocabulary adheres. The conciseness of the SVG syntax is clearly displayed on the following lines, where we draw a red rectangle with its 1 mm wide sides colored red and its surface painted yellow (`rect` element on lines 7–8). Finally, the `text` element (lines 10–13) typesets the string in black using Helvetica at a size of 10 pt.

PDF nor SVG can perform calculations, i.e., all dimensions, coordinates, etc. must be pre-calculated by an external program and fed into the PDF or SVG source (e.g., lines 17, 43, 47, and 54 in the PDF instance and lines 4, 7, and 10 in the SVG instance of our example). On the other hand, the PostScript interpreter when reading the PostScript instance calculates the size of the rectangle by measuring the width of the text string at the specified point size in the given font. Moreover, we decided to use a font (*Helvetica*) which is recognized as part of the “basic system” by the programs which interpret

<sup>1</sup>The use of this entry improves an application’s chances of finding the intended file and allows it to warn the user if the file has changed since the identifier was created[3, Section 10.3 *File Identifiers*].

PostScript, PDF, an SVG. If we were to use a non-sasic font or non-encoded special characters, the font would have to be declared, included and re-encoded. This would have made the examples much more complex.

Finally, the PostScript, PDF, or SVG output generated by the utility programs described in this chapter (e.g., dvips, ps2pdf, dvi2svg) is much more complex than the code shown in this section. This is because in their preamble they define a lot of generic commands and functions (abbreviations, font definitions, etc.) that they use throughout the output file in order to minimize its size and optimize its performance on interpreters and viewers.

## 22.2 DVI to PostScript drivers and dvips

All current implementations of T<sub>E</sub>X include a DVI-to-PostScript driver or generate PostScript or PDF directly. The possibilities in this area of some of the more notable commercial and public-domain programs are reviewed below.

- The driver in Textures (<http://www.bluesky.com/products/textures.html> for the Macintosh supports virtual fonts, direct PostScript inclusion and all the functionality needed by the E<sub>T</sub><sub>X</sub> graphics package. Textures takes special care to provide facilities for high-end prepress work, with careful attention to the details of the Adobe *Document Structuring Conventions*<sup>1</sup> and the needs of color work.
- Andrew Trevorow's shareware Macintosh T<sub>E</sub>X package, OzTeX (see <http://www.trevorow.com/oztex>), has support for PostScript Type 1 and TrueType fonts, and includes dvips, whose `\special` it recognizes for its on-screen rendering.
- The drivers in Y&Y T<sub>E</sub>X (a matched pair of dvipsone and the Windows previewer dviwindo previewer, see <http://www.tug.org/yandy/>) are unique in that they rely totally on Adobe Type Manager (ATM) for display, and support PostScript Type 1 fonts directly, without the need for font metric files, since they obtain metric information directly from the outline font file. The drivers are strong on font re-encoding facilities but do *not* support virtual fonts and they provide partial font downloading.
- Michael Vulis' VTeX (*Visual T<sub>E</sub>X*, see <http://www.micropress-inc.com/>) is a T<sub>E</sub>X system that on Microsoft Windows offers native support for PostScript, SVG, PDF, HTML, as well as Open-Type and PostScript Type 1 font manipulation from inside the T<sub>E</sub>X engine.
- *Personal T<sub>E</sub>X*'s PCTeX (<http://www.pctex.com>) is an integrated system on Microsoft Windows that can generate PostScript and PDF directly, has full support for PostScript Type 1 and TrueType fonts, and knows about dvips `\special` commands. It comes with the nicely crafted *MathTime Pro* fonts.

In choosing a program for preparing your graphics you should take into account the importance of *portability*, and therefore use as much as possible the standard graphics and color packages that we describe in this book. In any case the output drivers should conform to standard for Encapsulated PostScript, or else include raw PostScript in the output that a DVI-to-PostScript driver understands. The list above details the functions supported by different drivers and is a good starting point for readers considering which system or driver to use. The most widely used DVI driver today is undoubtedly Tom Rokicki's dvips, which we describe next.

<sup>1</sup>See [http://partners.adobe.com/public/developer/en/ps/5001.DSC\\_Spec.pdf](http://partners.adobe.com/public/developer/en/ps/5001.DSC_Spec.pdf).

### 22.2.1 The dvips PostScript driver

For over a decade Tom Rokicki's dvips DVI-toPostScript driver has been rightly regarded as the standard by which other drivers are measured. Being a very mature product it is now a standard part of the T<sub>E</sub>X Live distribution and thus available for all current computer platforms (including Linux, Macintosh, and Microsoft Windows). The dvips program has many important and useful features, in particular its support for `\special` commands is extensive (MetaPost, emT<sub>E</sub>X, and tpic, colors).

Two features of dvips are particularly useful:

- *Automatic generation of missing fonts*: if all else fails, dvips temporarily halts and calls another program to build what is needed. This facility is configurable and is not limited simply to running MetaFont (for the automatic generation of fonts a script, e.g., on Linux MakeTeXPK is normally called).
- *Named configuration files*: almost all settings can be changed in configuration files that can be specified on the command line. This makes it easy to control several devices with different characteristics.

### 22.2.2 Command line and configuration file options

The output from dvips can be controlled in two ways: by command line switches for a particular job and by commands in one or more configuration files, which let you set parameters globally for the whole system, on a per-printer basis, and on a per-user basis.

When dvips starts up, a global `config.ps` file is searched for (this must exist; the search path for configuration files depends on how the program was compiled). After this master file is loaded, Unix versions of dvips try to load a configuration file called `.dvipsrc` from the user's home directory, and other systems look for a file called `dvips.ini`. Next the command line is read. If the `-P` option is used, the corresponding configuration file is read at this point. Each configuration file (there can be multiple `-P` options) can override anything in the global or user configuration file, and it can also override anything seen in the command line up to the point that the current `-P` option is read.

After the command line has been completely scanned, if no `-P` option was selected, and the `-o` and `-f` options were not set, a `PRINTER` environment variable is searched for. If this variable exists and a configuration file for the corresponding printer exists, this configuration file is loaded last of all. Since the configuration path usually includes the current directory and can be set to include any of the user's directories, users can provide local versions of printer-specific configuration files to override any global system copies.

The list of all command line options is displayed by calling dvips as follows.

```
> dvips --help
Usage: dvips [OPTION]... FILENAME[.dvi]
a*  Conserve memory, not time      A   Print only odd (TeX) pages
b # Page copies, for posters e.g.  B   Print only even (TeX) pages
c # Uncollated copies              C # Collated copies
d # Debugging                     D # Resolution
e # Maxdrift value                 E*  Try to create EPSF
f*  Run as filter                   F*  Send control-D at end
                                   G*  Shift low chars to higher pos.

h f Add header file
i*  Separate file per section
j*  Download fonts partially
k*  Print crop marks               K*  Pull comments from inclusions
l # Last page
```

```

m* Manual feed                      M* Don't make fonts
mode s Metafont device name
n # Maximum number of pages         N* No structured comments
noomega Disable Omega extensions
o f Output file                     O c Set/change paper offset
p # First page                      P s Load config.$s
pp l Print only pages listed
q* Run quietly
r* Reverse order of pages           R* Run securely
s* Enclose output in save/restore S # Max section size in pages
t s Paper format                    T c Specify desired page size
u s PS mapfile                      U* Disable string param trick
v Print version number and quit      V* Send downloadable PS fonts as PK
x # Override dvi magnification       X # Horizontal resolution
y # Multiply by dvi magnification     Y # Vertical resolution
z* Hyper PS                         Z* Compress bitmap fonts
# = number    f = file    s = string  * = suffix, '0' to turn off
c = comma-separated dimension pair (e.g., 3.2in,-32.1cm)
l = comma-separated list of page ranges (e.g., 1-4,7-9)

```

Email bug reports to [tex-k@mail.tug.org](mailto:tex-k@mail.tug.org).

The number of command-line switches (and configuration options) is rather large (in fact most functions can be accessed in either way). The following catalogue classifies the available functions by command line switch and/or the configuration file command letter. Most options with no parameter can be turned off by immediately suffixing the option with a zero.<sup>1</sup> Within a configuration file, any empty line or line starting with a space, asterisk, equal sign, or a pound sign is ignored. All other lines must take the form of a command letter, followed by a space, and then some parameter.

## Page selection

**-p**  *number* Start printing at the page numbered *number*. The default is the first page in the document. If *number* is prefixed by =, then it is treated as the absolute sequence number rather than the  $\text{\TeX}$  page number (technically, the value of `\count0`). Thus, using `-p=9` starts with the ninth page of the document, regardless of how the pages are actually numbered on the printout.

**-l**  *number* Stop printing after the page numbered *number*. The default is the last page in the document. A prefix of = means that the number is relative to the number of pages in the file rather than the  $\text{\TeX}$  page numbers.

**-pp**  *numbers* Print the *range* of pages specified by a comma-separated list of *numbers*. This list can consist of individual page numbers or page ranges of two numbers separated by a colon. Thus you can print pages 3 to 10, 21, and 73 to 92 by specifying the option `-pp 3:10,21,73:92`; note that you cannot use this to reverse the sequence of pages—if you say `-pp 3,2,1`, pages 3, 2, and 1 are printed in their natural order.

**-n**  *number* Print *number* pages; the default is 100000, e.g., to get 20 pages starting from page 53, say `-p 53 -n 20`.

**-A**  Print only the *odd-numbered* pages.

**-B**  Print only the *even-numbered* pages.

<sup>1</sup>For instance, to turn off page reversal if it is on by default, use `-r0` on the command line or `r0` in a configuration file. The options that can be turned off in this way are `-a`, `-E`, `-f`, `-F`, `-G`, `-i`, `-j`, `-k`, `-K`, `-m`, `-M`, `-N`, `-q`, `-r`, `-R`, `-s`, `-U`, `-V`, `-z`, and `-Z`. They are signaled with a \* in the output of the command `dvips --help`.

**-r** **r** Produce pages in *reverse order*; by default, page 1 is printed first. This option is useful on printers like HP Deskjets that stack paper face up, so that the page printed first is at the bottom of the stack.

## Output

**-D** **D** *number* Set the the horizontal and vertical *resolution* in dpi (dots per inch) to *number*. This affects the choice of bitmap fonts that are loaded and also the positioning of letters in resident PostScript fonts. The number must be between 10 and 10000.

**-f** **f** Write the PostScript to *standard output*. Note, incidentally, that dvips requires the *.dvi* file to be “seekable”, so the input cannot come from a pipe.

**-m** Specify *manual feed* for the printer.

**-o** **o** *name* Send the PostScript *output* to a file called *name*; if no file name is given, then for an input file called *ffe.dvi*, the output file is called *ffe.ps*. If the first character of *name* is an exclamation mark, then the remainder is used as a command to pipe the output; e.g., if you specify **-o !lpr**, the output is fed to the standard Unix print command. (This may not be enabled under some operating systems.) The option also disables the use of the `PRINTER` environment variable and turns off the automatic sending of control-D if it was turned on with the **-F** option or in the configuration file; use **-F** *after* this option if you want both.

**-O** **O** *offset* Move the *origin* of the page. The *offset* is a comma-separated pair of *x, y* values such as *.1in, -.3cm*; the *x* value moves the page to the right and the *y* value moves it downwards. The default origin is one inch down and one inch to the right from the upper left corner of the paper.

**-t**  *papertype* Set the type of the paper to *papertype*. The value of *papertype* should be defined in one of the configuration files, along with the appropriate code to select it. You can also specify **-t landscape**, which rotates a document by 90 degrees. To rotate a document whose size is not letter, you can use the **-t** option twice, once for the page size and once for *landscape*. The upper left corner of each page in the *.dvi* file is placed one inch from the left and one inch from the top. Use of this option is highly dependent on the configuration file, since this is where paper types are defined.

**@** Define a paper size in a configuration; described in detail in Section 22.2.3.

**-T**  *offset* Set the size of the paper to the pair of dimensions *offset*. This option takes its arguments in the same style as **-O**. It overrides any paper size `\special` in the *.dvi* file.

**-x**  *number* Set the *magnification ratio* to *number*/1000 (*number* must be between 10 and 100000, and can be a real number if increased precision is desired), overriding the magnification specified in the *.dvi* file. It is recommended to use standard magstep values (1095, 1200, 1440, 1728, 2074, 2488, 2986, and so on) to limit the total number of *.pk* files generated.

**-X** **X** *number* Set the *horizontal resolution* only in dots per inch to *number*.

**-y**  *number* Set the *magnification ratio* to *number*/1000 times the magnification specified in the *.dvi* file (see **-x** above).

## Output configuration

**-E**  Attempt to *generate Encapsulated PostScript* with the correct bounding box. This works only on one-page files (or one-page selections from longer files), and it considers only the positions of characters and rules, ignoring any included graphics (see p. 133 for a description of the `epstool` program to correct the bounding box information, if needed). In addition, it gets the glyph metrics from the *.tfm* file, so characters lying outside their enclosing *.tfm* box may confuse it; Note also



that dvips output may be resolution-dependent if .pk fonts are used and this does not make very good EPS files, especially if the images are to be scaled.

**-e** **e** *number* Place each character so that it is no more than *number* pixels from its “true” resolution-independent position on the page. The default value of this parameter is resolution-dependent (it is the number of entries in the list 100, 200, 300, 400, 500, 600, 800, 1000, 1200, 1600, 2000, 2400, 2800, 3200, ... that are less than or equal to the resolution in dots per inch). Allowing individual characters to “drift” from their correctly rounded positions by a few pixels and then regain the true position at the beginning of each new word improves the spacing of letters within words. However, this facility can be dangerous in some tabular work (e.g., with the `hline` package) with complicated rules that may not be aligned properly. If you encounter problems, set the value to 0.

**-h** **h** *name* Insert file *name* as an additional *PostScript header*. If the name is simply “-”, this option suppresses the addition of all header files to the output. The contents of the header file is inserted into the PostScript `userdict` dictionary.

**-k**  Print *crop marks*. This option increases the paper size (which should be specified, either with a paper size `\special` or with the `-T` option) by half an inch in each dimension. Each page is then moved by a quarter of an inch and crop marks are drawn in each corner.

## Font handling

**--mode** **M** *mode* Specify the MetaFont *mode* for any .pk fonts that may have to be created. If this contradicts a specified resolution, it is ignored.

**-M**  Turn *oTtautomatic font generation*. If any fonts are missing, commands to generate the fonts are appended to the file `missfont.log` in the current directory.

**-V**  Download nonresident PostScript fonts as bitmaps. This requires use of programs like `ps2pk` or `gsftopk` (both part of the T<sub>E</sub>X Live distribution) to generate the required bitmap fonts.

**-Z** **Z** Compress bitmapped fonts before they are downloaded, thereby reducing the size of the PostScript file. This is very useful at high resolutions or when very large fonts are used, but it slows down printing, especially on older printers.

**-j** **j** Partially download PostScript Type 1 fonts. Font subsets are created containing only those characters used in the document.

**R** *numbers* List of default resolutions (the entries in the list, *numbers*, should be sorted in increasing order and separated by spaces). These resolutions are used to search for .pk fonts, if the size requested is not available. The output then uses PostScript scaling to scale the font up to the requested size. The resultant output is ugly, and a warning is issued. To turn this feature off, use a line containing `R` and no numbers.

## The configuration file and the printer

**-P**  *configname* Set up the output for a particular *configuration* or printer. This is implemented by reading in `config.confname`, which can set most of the options. It is recommended that all standard defaults go in one master `config.ps` file and that only things that vary from printer to printer go in the configuration files, as `config.ps` is read before `config.confname`. If no `-P` command is given, the environment variable `PRINTER` is checked. If that variable exists and a corresponding configuration file is found, that latter file is read. Alternatively (under Unix only), specifying a `-P` option with no corresponding configuration file is interpreted as a request to send the output to a system printer called *configname*. Thus `-Plp1` would direct output to the system `lp1` printer queue if `config.lp1` does not exist.

`-m` *number* Set *number* (default 180000) as the size of *printer memory* that can be used for fonts. This is intended for memory conservation and automatic splitting of the PostScript output into sections.<sup>1</sup> For typical T<sub>E</sub>X jobs, you should worry if the reported memory size is less than about 300000. If you have “unlimited” virtual memory, just use a very high number like one million.

`-I` Ignore the PRINTER environment variable.

`-p` *name* Set the name of the file (default `psfonts.map`) which lists available PostScript fonts. This option can distinguish between different sets of resident fonts in printers. If the name starts with a “+” character, then the rest of the name is used as an *additional* map file, rather than one overriding the current set.

`-u` `psmapfile` Set `psmapfile` to be the file that dvips uses for looking up PostScript font aliases. If `psmapfile` begins with a “+” character, then the rest of the name is used as the name of the map file, and the map file is appended to the list of map files (instead of replacing the list). In either case, if the name has no extension, then `.map` is added at the end (similar to the `p` command in `config.ps`).

## Memory considerations

`-a` `a` Conserve memory by making three passes over the `.dvi` file instead of two in order to load only those characters actually used.

`-i` Output each section into a separate file. Under certain circumstances, dvips splits the document up into “sections” to be processed independently. With this option, dvips places each section into a separate file; the new file names are constructed by replacing the suffix of the supplied output file name by a three-digit sequence number. This option is most often used in conjunction with the `-s` option, which sets the maximum section length in pages. If, for instance, your device cannot handle more than 10 pages at a time in one job, you can use `-i -s 10` to split it automatically into ten-page sections that are written into separate files.

`--noomega` Disable the use of Omega extensions when interpreting DVI files. By default, the additional opcodes “129” and “134” are recognized by dvips as Omega extensions and interpreted as requests to set two-byte characters, which means that the virtual font array will (temporarily) require 65536 positions instead of the default 256 positions, and make the memory requirements of dvips slightly larger. If this is unacceptable this option switches the extension off.

`-s` `i` *number* Set the *maximum number of pages* in each “section”. This option is most commonly used with `-i`; when used in a configuration file, the `i` option is equivalent to `-i -s`.

## Miscellaneous

`-b` `b` *number* Generate *number* copies of each page by duplicating the page body. This can be used, for example, in conjunction with a header file setting `bop-hook` to do color separations or to typeset posters.

`-c` *number* Generate *number* copies of every page by using PostScript’s `#copies` feature. Default is 1 (for collated copies, see the `-C` option).

`-C` *number* Create *number* of *collated* copies (by replicating the data in the PostScript file).

<sup>1</sup>The available memory can be queried by sending a file containing the following to your printer:

```
%!PS-Adobe-3.0
/Times-Roman findfont 30 scalefont setfont 144 432 moveto
vmstatus exch sub 40 string cvs show pop showpage
```

It is usually a good idea to tell dvips that the printer has rather *less* memory than the number returned by this procedure, as other things may fill up memory.



This takes longer to print than with the `-c` option, but it is much more convenient and is easier than submitting the same PostScript file over and over again.

- `-d`  *number* Set *debugging* options. This is intended only for emergencies or detailed examination by experts; it works only if dvips has been compiled with the `DEBUG` option. The useful values of *number* are discussed in Section 22.2.7; use a value of `-1` for maximum output.
- `--help`  Print the usage message and exit (see page 98).
- `-q`  *q* Run in *quiet mode*. Normally, dvips shows page numbers and the names of files that are included; this option tells it to print nothing but error messages to the normal error output.
- `-R`  *z* Run *securely*. This disables command execution in `\special` (via ```) and configuration files (via the `"E"` option), pipes as output files, and opening of any absolute filenames. The `-R1` option overrides a `z` command in the config file.
- `--version`  Print the dvips version number and exit.
- `-v`  Identical to `--version`.
- `-z`  Pass "hypertext" specials through to the output for eventual distillation into PDF (with Adobe Acrobat or ps2pdf, see page 131). The `-z` option is not enabled by default. With the `-z` option active, dvips will convert `\special:html...` commands into the corresponding `pdfmark` commands, otherwise they will be ignored.
- `E` *command* Execute the system *command* immediately, while the configuration file is being read. In many installations this is disabled so as to simply produce an error message, as it is clearly a security risk.
- `W` *text* If *text* is given, it is written to the standard (error) output; just the option by itself cancels any previous message. This is useful in the default configuration file if you want to require the user to specify a printer, for instance, or if you want to notify the user that the resultant output has special characteristics.

## Paths

- `H` *path* Search *path* for PostScript header files (including PostScript Type 1 fonts). The environment variable `DVIPSHEADERS` overrides this.
- `P` *path* Search *path* for bitmap `.pk` font files. If `%` characters are found in *path*, substitutions as described below are made, depending on the following letter, and a search is then undertaken for the resulting filename.

<code>%f</code>	font name
<code>%b</code>	output device horizontal resolution (dots per inch)
<code>%d</code>	font size (dots per inch)
<code>%p</code>	font type (this is always <code>.pk</code> )
<code>%m</code>	font mode (given by the <code>M</code> option)

If a path *does* contain a `%` character, the full filename must be given, including the path, rather than just the directory name; a path element such as `/fonts/%b` tries to open `/fonts/300` when looking for `cmr10.329pk`, for instance, and this may not be what is intended; `/fonts/%b/%f.%dpk` is needed.

- `S` *path* Search *path* for illustrations (such as Encapsulated PostScript files).
- `T` *path* Search *path* for `.tfm` files.
- `V` *path* Search *path* for virtual font `.vff` files.

## Compatibility issues

**-F** ☐ Add Control-D (ASCII code 4) as the very last character of the PostScript file. This is useful with an operating system interface that talks to the printer directly instead of via spooling software. The Control-D indicates to some PostScript interpreters that a job is finished.

**-G** ☐ Shift non-printing characters (ASCII codes 0–32, mostly control characters, and 127) to higher-numbered positions. This may be useful sometimes.

**-K** ☐ **K** Remove comments when including PostScript graphics, font files, and headers. This is sometimes necessary to get around bugs in spoolers or PostScript post-processing programs. Specifically, %%Page comments inside EPS files often cause difficulties. However, use of this option can cause some included graphics to fail, since the PostScript header macros from some software packages read portions of the input stream line by line, searching for a particular comment. The option is turned *on* by default because some PostScript previewers and spoolers have problems with the structuring conventions.

**-N** ☐ **N** Turn off *structured comments*; this might be necessary on some systems that try to interpret PostScript comments in the wrong way, or on some old PostScript printers or software.

**-S** ☐ **S** Enclose the entire output in a PostScript *save* and *restore* pair. Try to avoid using this, since the result is not proper EPS, but it may be necessary if you are driving the printer directly and do not care about the portability of the output.

**-U** ☐ **U** Disable a PostScript virtual memory saving optimization that stores the character metric information in the same string used to store the bitmap information. This is only necessary when using a Xerox 4045 PostScript interpreter, which has an odd bug.

All of the path-searching configurations can be overridden at run-time by setting environment variables (see the program documentation for more details).

### 22.2.2.1 The standard dvips configuration file

The T<sub>E</sub>X Live distribution comes with the following `config.ps` configuration file.

```

1  % teTeX's config.ps. Thomas Esser, 1998, public domain.
2
3  % Memory available. Download the three-line PostScript file:
4  %   %! Hey, we're PostScript
5  %   /Times-Roman findfont 30 scalefont setfont 144 432 moveto
6  %   vmstatus exch sub 40 string cvs show pop showpage
7  % to determine this number. (It will be the only thing printed.)
8  m 3500000
9
10 % z1 is "secure", i.e., inhibits execution of 'shell commands' in
11 % \specials. Dvips allows this by default.
12 z1
13
14 % How to print, maybe with lp instead lpr, etc. If commented-out, output
15 % will go into a file by default.
16 %o |lpr
17
18 % Default resolution of this device, in dots per inch.
19 D 600
20 X 600
21 Y 600
22
23 % Metafont mode. (This is completely different from the -M
24 % command-line option, which controls whether mktexpk is invoked.)
25 % See ../../metafont/misc/modes.mf for a list of mode names. This mode
26 % and the D number above must agree, or mktexpk will get confused.
27 M ljfour
28
29 % Last resort bitmap sizes.
30 R 300 600

```

```

31
32 % Correct printer offset. You can use testpage.tex from the LaTeX
33 % distribution to find these numbers.
34 O 0pt,0pt
35
36 % Bitmap font compression. Results in more compact output files, but
37 % sometimes causes trouble. So the default is disabled. Set Z1 to enable
38 % this feature.
39 Z0
40
41 % Partially download Type 1 fonts by default. Only reason not to do
42 % this is if you encounter bugs. (Please report them to
43 % @email{tex-k@mail.tug.org} if you do.)
44 j
45
46 % This shows how to add your own map file.
47 % An "all-in-one" psfonts.map.
48 p psfonts.map
49
50 % Instead of psfonts.map, you can use smaller "modules".
51 % See updmap script for how they are concatenated to make psfonts.map
52
53 % To use the CM Typel fonts
54 % p +bsr.map
55 % "real" bakoma instead of interpolated bsr
56 % p +bakomaextra.map
57 % this one *or* the previous one. Not both!
58 % p +bsr-interpolated.map
59 % Taco Hoekwater's additions
60 % p +hoekwater.map
61
62 % 0 0 595 842 is the right bounding box that most applications expect
63 % for A4. Since dvips always rounds up, choose something slightly smaller.
64
65 @ A4size 594.99bp 841.99bp
66 @+ ! %%DocumentPaperSizes: a4
67 @+ %%PaperSize: A4
68
69 @ letterSize 8.5in 11in
70 @+ ! %%DocumentPaperSizes: Letter
71
72 .... MANY ENTRIES NOT SHOWN ....
73
74 @ unknown 0in 0in
75 @+ statusdict /setpageparams known { hsize vsize 0 1 statusdict begin {
76 @+ setpageparams } stopped end } { true } ifelse { statusdict /setpage known
77 @+ { hsize vsize 1 statusdict begin { setpage } stopped pop end } if } if

```

After the explanation of the configuration file options on the previous pages most of the entries should be self-evident. The page size definitions (lines 65–77) are explained in the next section. In fact we only show the first two entries that are present in the file installed by T<sub>E</sub>X Live.<sup>1</sup> It is important to note that the first page size in the file (A4size on line 65) will be used as the default if no explicit page size is specified when running dvips. The last four lines (74–77) are similar to the definitions needed for the linotype (see p. 107). For more information about the the font-related map entries (lines 46–60) see Section 22.2.5.

### 22.2.3 Paper sizes

Documents are normally designed for a particular paper size, whether in portrait or landscape orientation. Information on page size should ideally be placed in the .dvi file, not chosen at print time, and dvips therefore supports a `papersize \special` command, although it can also be specified in configuration files and on the command line. The format of the `\special` command (which must occur

<sup>1</sup>The full list of page sizes defined is: letter, legal, ledger, tabloid, a6, a5, a4, a3, a2, a1, a0, b6, b5, b4, b3, jisb0, jisb1, jisb2, jisb3, jisb4, jisb5, jisb6, jisb7, and jisb8, see also Table 22.1

somewhere on the first page of the document)<sup>1</sup> is:

```
\special{papersize=dimension,dimension}
```

The first dimension is the horizontal size of the paper and the second is the vertical size; they can be expressed using any of the normal T<sub>E</sub>X dimensions of *in* (inches), *cm* (centimeters), *mm* (millimeters), *pt* (points), *sp* (scaled points), *bp* (big points, the same as the default PostScript unit), *pc* (picas), *dd* (didot points), and *cc* (ciceros). Thus American letter-size paper would be chosen by putting:

```
\special{papersize=8.5in,11in}
```

in the T<sub>E</sub>X document (or, better, in the document class), and A4 paper to be used in landscape would be chosen with:

```
\special{papersize=297mm,210mm}
```

Of course, using such a command merely informs dvips of the desired paper size; you must set the `\textwidth` and `\textheight` dimensions (and the appropriate margins) in your T<sub>E</sub>X document class to actually make use of the page size.

When dvips processes a file, it matches up the requested paper size with a setup defined in the configuration file, from which it works out what to put in the PostScript output. Paper sizes are defined in configuration files by lines starting with @.

There are three formats for @ lines. First, an option specified on a line by itself, with no parameters, instructs dvips to discard all other paper size information (possibly from another configuration file) and start from scratch. Second, if the option contains three parameters, a keyword followed by two dimensions, it is interpreted as starting a new paper size description, where the keyword is the name and the dimensions are the horizontal and vertical size of the sheet of paper. Third, if both dimensions are zero (note that you must still supply units, like 0mm), then any page size in the .dvi file is a match. If the @ character is immediately followed by a + character, then the remainder of the line (after skipping any leading blanks) is treated as PostScript code to send to the printer to select the paper size being defined. Within those + lines, if the first character is an exclamation mark, then the line is put in the initial *comments* section of the final output file; otherwise, it is put in the *setup* section of the output file—thus the definition of A4 paper in the standard configuration file looks like this:

```
@ A4 210mm 297mm
@+ ! %%DocumentPaperSizes: A4
@+ %%BeginPaperSize: A4
@+ a4
@+ %%EndPaperSize
```

This indicates that the structured comment `%%DocumentPaperSizes: A4` is to be placed in the initial document description area, and the `PaperSize` lines go in the setup section. In this example, the single PostScript command `a4` (defined in the dvips header files) is what actually does the work of instructing the PostScript interpreter, but the code may be arbitrarily complicated. If you know that all your printers understand Level 2 PostScript, you can use a command like:

```
@+ << /PageSize [ 595 842 ] >> setpagedevice
```

When dvips sees a paper format on the command line, it looks for a match by *name*; when it sees a `papersize \special` command, it looks for a match by dimensions (they must match within a quarter of an inch). The first match found (in the order in which the paper size information is found

<sup>1</sup>T<sub>E</sub>X has a convenient way of doing this with the `\AtBeginDvi` macro.

in the configuration file) is used. If nothing matches, a warning is printed and the first paper size given is used, so the first paper size should always be the default.

Landscape mode for all paper sizes is supported automatically; after trying to match a pair of dimensions, dvips then tries to match them in reverse. If that succeeds, it inserts PostScript code in the output to rotate the pages. Note that this is not the same as including instructions asking the printer to switch to a different paper tray.

If our  $\text{\LaTeX}$  file contains

```
\special{papersize=297mm,210mm}
```

dvips finds that a paper size of 210mm  $\times$  297mm in the configuration file makes a match, but adds landscape mode. By default it behaves as if the paper were rotated ninety degrees counterclockwise; if your landscape orientation should be rotated clockwise, the document class or document should include the following code:

```
\special{! /landplus90 true store}
```

(`\special` commands like this are discussed in Section 22.2.4). However, if the printer directly *supports* landscape A4 paper, then the configuration file should have a special entry for 297mm  $\times$  210mm and supply the correct PostScript code to select the device.

If your printer has a command to set special arbitrary paper sizes, then specify dimensions of size zero (like 0in 0in); the PostScript code that sets the paper size can refer to the user-requested dimensions as `hsize` and `vsize` which are macros defined in PostScript that return the requested size in default PostScript units. For example, in sending output to a Linotronic typesetter, which has a continuous roll of paper, you can define a paper size name to make the text run across the width of the film as follows:

```
@ film1 0in 0in
@+ statusdict /setpageparams known
@+ { hsize vsize 0 1 statusdict begin
@+ { setpageparams } stopped end }
@+ { true }
@+ ifelse
@+ { statusdict /setpage known
@+ { hsize vsize 1 statusdict begin
@+ { setpage } stopped pop end } if
@+ }
@+if
```

This could then be used with a command line option of `-t film1 -T 297mm,420mm` to set A3 pages with the long side along the edge of the film. dvips is distributed with a configuration file `config.lino` that defines such setups.

Note that the PostScript commands used to define paper sizes may well be device dependent and thus make the output less portable.

### 22.2.4 Interaction with PostScript

It is strongly recommended to *not* directly use dvips `\special` commands in your  $\text{\LaTeX}$  documents, but instead use higher-level macro packages that provide portable interfaces. In particular, all graphics inclusion, rotation, scaling, and color typesetting should be performed using the standard  $\text{\LaTeX}$

graphicx and color packages. The lower-level functions briefly described in this section are merely intended to give writers of new macro packages and those curious about how the packages work under the hood a hint of what is available.<sup>1</sup>

The wide range of `\special` commands supported by dvips fall into several groups:

1. specifying resources, general configuration with extra definitions (e.g., header files);
2. inserting PostScript figures, in particular EPS files;
3. inserting literal PostScript code;
4. defining and using color;
5. producing HyperPostScript to make PDF;
6. supporting the conventions of `emTeX`, `tpic`, and `MetaPost`.

#### 22.2.4.1 Including PostScript files and literal PostScript code

The `\special` commands to steer the inclusion of resources and graphics are the following:

Name	Example	Comments
<code>papersize</code>	<code>\special{papersize=5in,8in}</code>	set paper size
<code>landscape</code>	<code>\special{landscape}</code>	special form of <code>papersize</code> to select landscape orientation of default paper size
<code>PSfile</code>	<code>\special{PSfile="pot.ps"}</code>	insert graphics file
<code>psfile</code>	<code>\special{psfile=pot.ps}</code>	insert graphics file (alternative syntax)
<code>header</code>	<code>\special{header=duplex.pro}</code>	insert PostScript header file into output

The `\special{PSfile=...}` command is followed by a set of *keyword=value* pairs (each pair separated by spaces) to specify the picture size and other parameters. The possible keys are similar to those of the `graphicx` package (see the dvips documentation). Moreover, dvips does *not* interpret itself the graphics file to be included to determine the bounding box, clipping, etc. so that in any case this information will have to be transmitted to dvips by the `graphicx` package.

To include some graphics files in dvips output or to support sophisticated packages that use literal PostScript in `TeX` (such as `PSTricks`), a certain font or header file might have to be sent first. Therefore, dvips provides the `\special{header=...}` syntax, whose general form is as follows:

```
\special{header={file.ps} pre={pre code} post=post code}
```

The header *file.ps* will be included in the output document, preceded by the PostScript code *pre code*, and followed by *post code*. This syntax requires the use of balanced braces in all arguments.

For instance, if the same logo is used several times in a document, it is appropriate to load it only once at the beginning of the document, e.g.,

```
\special{header=mylogo.eps}
```

If a non built-in font is used in one of the figures, it must be included separately. As an example a font instance of *Adobe Utopia Roman* can be included as follows:

```
\special{header=putr.pfb}
```

There are two ways to insert PostScript code directly into the output; in the first, “safe” method, the code is surrounded in the output by PostScript commands to limit its effect; the second method is unprotected and can affect the current drawing state. The latter is needed for operations like text

<sup>1</sup>The full documentation at <http://www.tug.org/texinfohtml/dvips.html> should be consulted for more details.

rotation, when we really *do* want to change the current state, while the former might be used to draw a page border independently of the main text.

The `\special` commands for the *safe* insertion of literal PostScript are:

Name	Example	Comments
"	<code>\special{"newpath 0 0 moveto} 100 100 lineto stroke</code>	insert literal PostScript
!	<code>\special{! /magscale false def}</code>	insert literal PostScript into user dictionary

You can define your own macros for use in literal graphics code; they are defined just like literal graphics, except that the `\special` command begins with an exclamation mark instead of a double quote. These literal macros are included as part of the header material in a special dictionary, `SDict`, which is the first one on the PostScript dictionary stack when any literal PostScript is used.

The `\special` commands for the *unprotected* insertion of literal PostScript are:

Name	Example	Comments
<code>ps:</code>	<code>\special{ps: .5 setgray}</code>	insert literal PostScript
<code>ps::</code>	<code>\special{ps:: 0.5 setgray}</code>	insert literal PostScript
<code>ps:[begin]</code>	<code>\special{ps::[begin]gsave .5 setgray}</code>	start inserting literal PostScript
<code>ps:[end]</code>	<code>\special{ps::[end]grestore}</code>	end inserting literal PostScript
<code>ps: plotfile</code>	<code>\special{ps: plotfile foo.ps}</code>	insert file into output

The `::` commands can be used to construct an unbroken set of PostScript commands in the output with a series of `ps::` commands, starting with `ps:: [begin]` and ending with `ps:: [end]`. This may be useful when you have a lot of PostScript to write. In general, however, use the simple `ps:` convention wherever possible. When writing unprotected PostScript you should realize that `dvips` works with its own coordinate system, which is dependent on the resolution (stored in the PostScript macro called `Resolution`).

Details about how `dvips` supports color and hypertext are in the `dvips` reference documentation.

## 22.2.5 Font support

Historically, `TEX` systems supplied `bitmap (.pk)` fonts to printers. Nowadays, the printing device most often supports PostScript, so that we can use PostScript Type 1 (or TrueType) variants of the `TEX` fonts, as well as built-in fonts. `dvips` manages the association between font names and PostScript font files with the help of the configuration file `psfonts.map`, which consists of lines that map a `TEX` font name onto a full PostScript font name. It has the form:

```
filename PostScript-name [options]
```

The simplest case somethinh like the following line, which instructs `dvips` to map the `TEX` name `bold-times` onto the full PostScript name `Times-Bold`:

```
boldtimes Times-Bold
```

More generally, `dvips` interprets the entries in the `psfonts.map` file as follows.

1. Empty lines or lines beginning with a space, percent, asterisk, semicolon, or hash mark are ignored.
2. Otherwise, the line is separated into “words”, defined as a group of characters separated by spaces, tabs, enclosed inside double quotes, or starting with a double quote and extending until the end of the line.

3. A word starting with “<” corresponds to a font file that is to be fully downloaded (i.e., partial downloading is prescribed).
4. A word starting with “<[” corresponds to an encoding file to be downloaded (file name ending in .enc). Note, however, that encoding files are often included just as header files (see next item).
5. A word starting with “<” corresponds to a header file to be downloaded. If the name ends in .pfa or .pfb, it is taken as a PostScript Type 1 font file (partially downloaded if the -j option is in effect). A font can have more than one header. If a “<” is a word by itself, the next word is taken as the name of the header file.
6. A word starting with a double quote (”) is interpreted as PostScript code to be used in generating the font instance. The code is inserted verbatim into the output at the appropriate point (the double quotes beginning and ending the word are removed).
7. Otherwise the word is a name. The first such name is the .tfm file that a virtual font file can refer to. If there is a second name, it is used as the PostScript name. If only a single name is specified it is used for both the T<sub>E</sub>X and the PostScript name.

When dvips finds a font name in the .dvi file it is checked against the list found in the map file `psfonts.map`. Usually font names follow the naming scheme described in Section 21.7 on page 67. Let us have a look at a few variants of the built-in *Palatino* font.

```

1 pplr8r Palatino-Roman "TeXBase1Encoding ReEncodeFont" <8r.enc
2 pplr8rn Palatino-Roman ".82 ExtendFont TeXBase1Encoding ReEncodeFont" <8r.enc
3 pplrr8re Palatino-Roman "1.2 ExtendFont TeXBase1Encoding ReEncodeFont" <8r.enc
4 pplro8r Palatino-Roman ".167 SlantFont TeXBase1Encoding ReEncodeFont" <8r.enc
5 pplru8r Palatino-Italic "-.1763 SlantFont TeXBase1Encoding ReEncodeFont" <8r.enc

```

These entries show two features of the mapping file described in the list above. First, (see item 6. above) the PostScript code inside double quotes is applied to the font when it is loaded. The string `TeXBase1Encoding ReEncodeFont` re-encodes the font to make all the characters available (lines 1–5) in a way T<sub>E</sub>X can use them. The PostScript commands `ReEncodeFont`, `Extendfont` (to apply horizontal scaling to squash the font to 82% of its normal width, as shown on line 2 or to extend it to 120% of its normal width, as shown on line 3) and `Slantfont` (line 4 shows how to slant an upright font 0.167 to the right creating an oblique variant, while line 5 shows how to slant an Italic font –0.176 to the left to create an upright variant) are defined in the header files that dvips downloads. Second, anything following a < is interpreted as a file name whose contents are to be placed in the output (list item 5. above); in this case the file `8r.enc` defines the `TeXBase1Encoding` encoding (see Sections 21.2.2 on page 23 and ff.).

The same < can also be used to instruct dvips to download a font with the current job to the printer and thereafter treat it as if it were built-in. The following line indicates that the font `cmr10.pfb` should be downloaded to satisfy requests for `cmr10` (PostScript name `CMR10`).

```
cmr10 CMR10 <cmr10.pfb
```

All the available features are exercised in the following two declarations, which are functionally (almost) equivalent:<sup>1</sup>

```

putro8r Utopia-Regular ".167 SlantFont TeXBase1Encoding ReEncodeFont"
                                         <8r.enc <putr8a.pfb

putro8r Utopia-Regular ".167 SlantFont TeXBase1Encoding ReEncodeFont"
                                         <[8r.enc <<putr8a.pfb

```

<sup>1</sup>These declarations should be on a *single* line in the `psfonts.map` file. We have split them between two lines for reasons of readability.



Both these lines map a slanted and re-encoded version of Utopia-Regular (downloaded with the job) to the T<sub>E</sub>X name `putro8r`. The first instance uses the global `<` notation to include files, while the second instance includes the encoding file with the more specific `<[` operator (see item 4. above). Therefore, the only practical difference between the instances is that `<<` (see item 3. above) specifies that we want the file `putr8a` to be fully downloaded, while the first declaration (using `<`) allows partial downloading, if it is requested. Note that font files (and encoding files) are downloaded only once per job, however often map files refer to them.

When installing the T<sub>E</sub>X Live distribution all the mapping lines for all fonts available on your system are entered into the default file called `psfonts.map`. However, dvips allows finer granularity by the use of multiple configuration files and map files. Extra map files can be loaded with the `p` command in a configuration file; thus

```
p new.map
```

establishes `new.map` as the *only* set of mappings, whereas

```
p +new.map
```

appends the values in `new.map` to the current list. There are some suggestions included as comments in the default dvips configuration file (p. 104ff, lines 46–60).

### 22.2.6 Special hooks

dvips provides a series of “hooks” that let the user plug in code to be executed at several useful places. It does this by defining four PostScript procedures (`start-hook`, `end-hook`, `bop-hook`, and `eop-hook`) in the PostScript `userdict`; these procedures are executed at the start and the end of the document and at the beginning and the end of every page, respectively. The procedures can be *redefined* in headers added by the `-h` option or the `\special{header=...}` command. This code is executed outside the `save/restore` context of the page, so it can accumulate information about the whole document (unless it is broken into sections because of memory constraints.) The default PostScript coordinate system and origin are in effect when the code in these procedures is run.

When `bop-hook` is executed, two values are placed on the PostScript stack, the T<sub>E</sub>X page number and the sequence number of the page in the file. For `start-hook`, the horizontal size, the vertical size, the magnification, the horizontal and vertical resolutions (in dpi), and the name of the `.dvi` input file are put on the stack. The procedures must *leave* these parameters on the stack. The other hooks have no values on the stack.

If you *do* use `bop-hook` or `eop-hook` to keep information across pages, then the document no longer complies with the Adobe document structuring conventions, so you should use the `-N` option to turn off structured comments.

Typical uses of the `bop-hook` procedure are writing very large words across each page (overlaying the text) or printing cropmark identifiers in the corner of each page. For example, if you put the following code into a file called `secret.pro`:

```
userdict begin
/bop-hook{gsave
  200 100 translate
  65 rotate
  /Times-Bold findfont 124 scalefont setfont
  0 0 moveto
  0.7 setgray
```

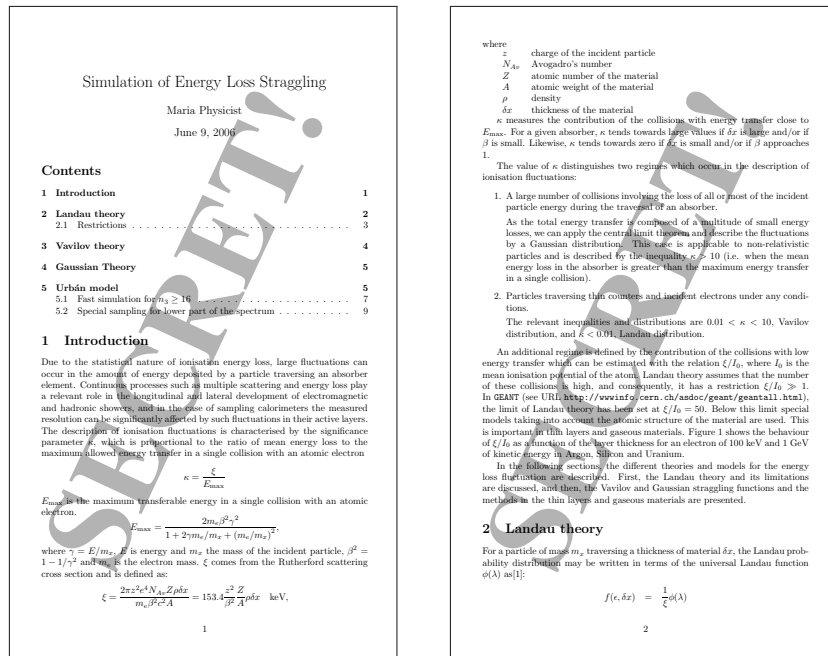


Figure 22.1: Stamping pages using dvips

```
(SECRET!) show
grestore} def end
```

and run a file called `test.dvi` through `dvips` with the command line

```
> dvips -h secret.pro test.dvi -o test.ps
```

each page looks like those in Figure 22.1.

Jürgen Vollmer’s `draftcopy` package uses this technique to offer a more sophisticated  $\LaTeX$  interface to add overlay information on pages; the default text can be in different languages (“Draft” in English, “Entwurf” in German, etc.), or the user can specify some other word. It also supports several DVI drivers other than `dvips`.

Sergio Callegari’s `draftwatermark` package takes a more lightweight approach, which does not rely on PostScript specials and is fully compatible with `pdflatex`. This `draftwatermark` package, which is based on the `everypage` package by the same author, lets you add a textual, light gray watermark on every page (default) or on the first page of a document only, by specifying the `firstpage` option, as follows

```
\usepackage[firstpage]{draftwatermark}
```

Five commands, each one taking a single argument, let you influence the way the watermark is presented.

```
\SetWatermarkAngleangle
```

real number specifying the angle in degrees (default 45.) at which the watermark text is drawn;

```
\SetWatermarkLightnessgraylevel
```

real number specifying the gray level (1. for white, 0. for black, default 0.8) of the

```

        watermark text;

\SetWatermarkFontSize $\textit{fontsize}$ 
        length specifying the font size (default 5 cm) for the watermark text;

\SetWatermarkScale $\textit{scale}$ 
        real number specifying the scaling factor (default 1.2) to be applied to the watermark
        text;

\SetWatermarkText $\textit{text}$ 
        text string specifying the watermark text to be printed (default DRAFT).

```

An elegant trick suggested by the author of dvips is to write a header file dynamically from a configuration file that can give you, for instance, access to the current date. You can put the following code into a file called `config.dte`:

```

E echo /bop-hook userdict begin \{ gsave > date.txt
E echo /Times-Roman findfont 10 scalefont setfont >> date.txt
E echo 10 10 moveto \((printed on `date`\) show grestore >> date.txt
E echo \} def end >> date.txt
h date.txt

```

and then say

```
> dvips -R0 -Pdte test.dvi -o test.ps
```

to write, at the bottom of each page, the date on which the document went through dvips. This works only on Unix systems (the program must be able to execute commands like `date` from within itself). We specify the `R0` option to override the `z1` entry (line 12 in the default configuration file, see p. 104ff), which would otherwise disallow the `E` option in our configuration file `config.dte`.

As an alternative, you can write material to the PostScript header area using the `\special{!...}` technique; you can therefore recode the last example as:

```

\special{!
  userdict begin
    /bop-hook {gsave
      /Times-Roman findfont 10 scalefont setfont
      10 10 moveto (typeset on \today) show grestore
    } def end
  }

```

This of course writes the date on which the document was typeset, not put through dvips.

Another use for this kind of hook is in preprocess applications. In using  $\text{\TeX}$  and dvips to prepare film directly for a printer, you might need to print it “face down”, as a mirror image. The following code in a header file achieves this simply with PostScript `translate` and `scale` operators.<sup>1</sup>

```

userdict begin
  /hmirror {hsize 0 translate -1 1 scale} def % horizontal mirror
  /vmirror {0 vsize translate 1 -1 scale} def % vertical mirror
  /bop-hook{hmirror} def % apply horizontal mirror for each page
  % /bop-hook{vmirror} def % apply vertical mirror for each page
end

```

<sup>1</sup>The `translate` operator in the `hmirror` macro consumes the value of the PostScript macro `hsize`, where dvips stores the page width. Similarly, in `vmirror` we use the PostScript macro `vsize` which dvips sets to the vertical page size.

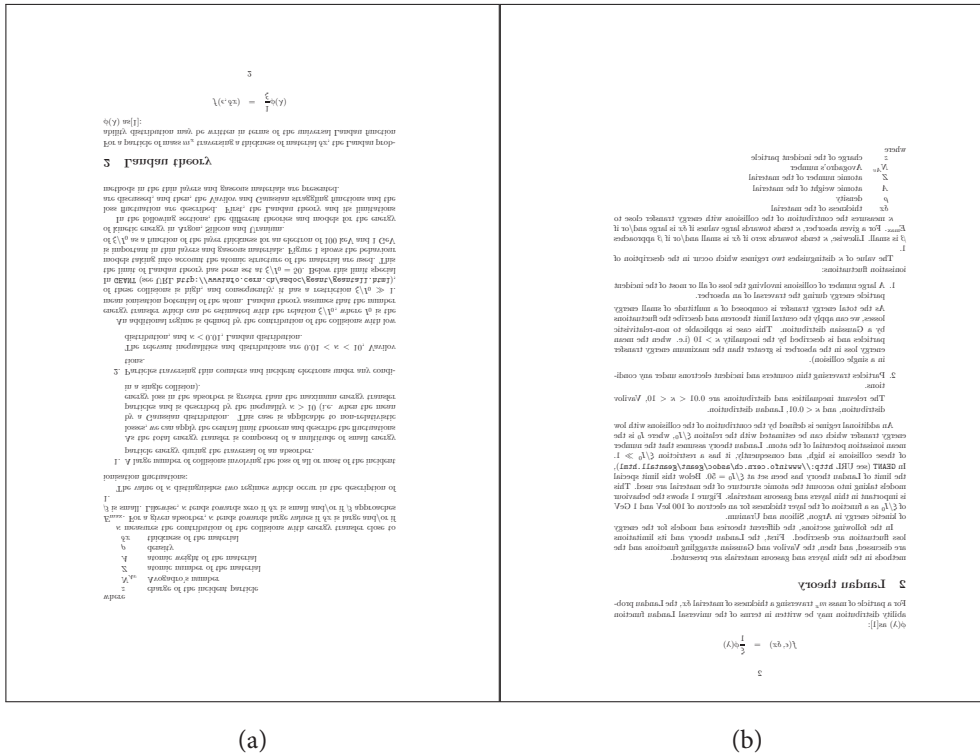


Figure 22.2: Mirror-image printing with dvips.

The effect of applying this header file is shown in Figure 22.2(a). A simple change (executing `vmirror` instead of `hmirror` in the `bop-hook` definition by commenting out line 4 and activating line 5 above) mirrors with respect to the horizontal axis (Figure 22.2(b)). In Level 2 PostScript this mirror effect, and the white-on-black effect also needed in printing, are directly supported with the `setpagedevice` command [see 2, p. 679]. This means you need merely insert the following code in the PostScript preamble:

```
<< /MirrorPrint true >> setpagedevice
```

to get mirroring, and

```
<< /NegativePrint true >> setpagedevice
```

to get black on white. Unfortunately, not all PostScript implementations support this yet, whereas high-end phototypesetters, for which these functions are important when printing to film, most often do.

Our final example shows how to display the current page number as an overlay on each page.

```
1 userdict begin
2   /mystring 20 string def
3   /center {% stack: string to be centered
4     dup stringwidth pop 2 div neg 0 rmoveto
5     } bind def
6   /bop-hook{% stack: page-number page-sequence-number
7     gsave
```

```

8           /Helvetica findfont 550 scalefont setfont
9           hsize 2 div 200 moveto
10          0.9 setgray
11          exch dup mystring cvs center show exch
12          grestore} def
13 end

```

After declaring the string variable `mystring` (line 2) a function, `center`, is defined which calculates the horizontal offset needed to center the string it receives on its stack (lines 3–5). As already mentioned, `dvips` puts the  $\TeX$  page number and page sequence number on the stack before executing `bop-hook`. Therefore, after setting the font to Helvetica (line 8) and defining the gray level (line 9) we are ready to add the  $\TeX$  page number as an overlay on each page (line 10). Therefore, we first have to get hold of the second entry on the stack (the page number in question) and move it to the top of the stack (`exch`), duplicate it (`dup`), since we have to leave the stack untouched, convert the number to a string (`cvs`) that is stored in `mystring`, which we center (`center`) and shown on the output medium (`show`). Finally the stack is put back in the correct order (`exch`, which undoes the effect of the first such command at the beginning of line 10). We used this header file to prepare the PostScript file that was used to create Figure 22.6 on page 141.

Color separations (e.g., making four separate instances of each page, corresponding to the base colors to be printed in four-color mode) can also be made with the `bop-hook` procedure. However, `dvips` makes no provision for checking the content or accumulating commands. Therefore if you want to combine several effects, e.g., write “SECRET!” on each page *and* print crop marks, you cannot just load two header files in succession, but you must write more sophisticated PostScript code. Bogusław Jackowski has contributed in the `macros/generic/TeX-PS` directory on CTAN a few example files where he shows how to combine such complex tasks.

### 22.2.6.1 Conversion between color models

It is possible to convert between color models by placing the relevant PostScript code inside a `\special` command to be included just before starting composing the body of the document (`\AtBeginDocument` command). The following example shows how to transform various color models into gray levels[2, Section 7.2 *Conversion among Device Color Spaces*, p. 473].

```

1  \AtBeginDocument{%
2  \special{ps:
3    /setcmykcolor { exch 0.11 mul add
4                   exch 0.59 mul add
5                   exch 0.3 mul add
6                   dup 1 gt { pop 1 } if neg 1 add setgray } def
7    /setrgbcolor { 0.11 mul
8                   exch 0.59 mul add
9                   exch 0.3 mul add setgray } def
10   /sethsbcolor { /b exch def /s exch def 6 mul dup cvi dup
11                 /i exch def sub /f exch def
12                 /F [[0 1 f sub 1][f 0 1][1 0 1 f sub][1 f 0]
13                   [1 f sub 1 0][0 1 f][0 1 1]] def
14                 F i get { s mul neg 1 add b mul } forall
15                 0.11 mul
16                 exch 0.59 mul add
17                 exch 0.3 mul add setgray } def
18   }
19 }

```

### 22.2.7 Debugging

You can use the `-d` option of `dvips` for tracking down errors and understanding what is going on. You must supply an integer specifying the class of information to be displayed, as follows:

1	specials	4	fonts	16	headers	64	files
2	paths	8	pages	32	font compression	128	memory

To get several types of information, simply add the numbers together for the types you are interested in. A value of `-1` selects them all. For example if you call `dvips` with the option `-d 4` you get something like this:

```
> dvips -d4 -nl exadvips
This is dvips(k) 5.95b Copyright 2005 Radical Eye Software (www.radicaleye.com)
' TeX output 2006.05.28:1655' -> exa.ps
Defining font () cmr17 at 17.3pt
Font cmr17 <CMR17> is resident.
Defining font () cmr12 at 12.0pt
Font cmr12 <CMR12> is resident.
Defining font () cmbx12 at 14.4pt
Font cmbx12 <CMBX12> is resident.
Defining font () cmbx10 at 10.0pt
Font cmbx10 <CMBX10> is resident.
Defining font () cmr10 at 10.0pt
Font cmr10 <CMR10> is resident.
Defining font () cmr10 at 10.0pt
Font cmr10 <CMR10> is resident.
Defining font () cmr10 at 10.0pt
Font cmr10 <CMR10> is resident.
Defining font () cmr7 at 7.0pt
Font cmr7 <CMR7> is resident.
Defining font () cmsy10 at 10.0pt
Font cmsy10 <CMSY10> is resident.
Defining font () cmr7 at 7.0pt
Font cmr7 <CMR7> is resident.
<tex.pro><texps.pro><special.pro>. <cmr7.pfb><cmsy10.pfb><cmr7.pfb>
<cmr10.pfb><cmr10.pfb><cmr10.pfb><cmr10.pfb><cmr10.pfb><cmr10.pfb><cmr10.pfb>[1]
```

This shows which fonts are needed for and how they are dereferenced (here PostScript Type 1 versions of the CM fonts are included in the output file and downloaded with the job).

## 22.3 Ghostscript, a PostScript interpreter

Aladdin Ghostscript (<http://www.cs.wisc.edu/~ghost/>) is a freely available PostScript interpreter written by L. Peter Deutsch. It can be used for various purposes:

- *Previewing PostScript code.* Ghostscript displays your PostScript or PDF on screen (supporting a wide variety of devices), making possible fast checking of your drawings before sending them to a printer; more sophisticated interfaces exist on Linux, Mac OS X, and Microsoft Windows (see Section 22.3.4).
- *Preparing output for various printing devices.* The Ghostscript program comes with a large set of drivers, so PostScript code can be transformed into a format that is understood by most printers (see Section 22.3.3).

- *Converting into raster formats.* Ghostscript can convert PostScript into many different raster formats, such as TIFF, PBM, and PNG (see Section 22.3.3).
- *Manipulating PostScript.* Ghostscript performs various helper tasks, such as extracting text from PostScript files, calculating the bounding box of an EPS file, and converting PostScript into Adobe Illustrator format (see Section 22.3.5).
- *Multi-platform PDF reader.* Ghostscript reads PDF (Portable Document Format) files, so that you can view Acrobat files (see Section 22.3.4). Ghostscript can “distill” PostScript code into PDF, or generate PostScript from PDF (see Section 22.3.5).

The source code of Ghostscript is mostly written in C and can be freely downloaded from the Internet and installed. Versions are available for all current computer platforms. How to start and use Ghostscript depends slightly on the operating system.<sup>1</sup> Nevertheless, the command-line to invoke Ghostscript is essentially the same on all systems, although the name of the executable program itself may differ. For instance, to invoke Ghostscript on Unix-like systems type:

```
gs [options] file1 ... [options] fileN ...
```

This will read each of the files and execute the PostScript commands contained in them. Depending on the options, the result may be displayed on screen, passed to some other process, or written to a file. When all files have been read, the program normally prompts for further keyboard input. To exit the interpreter, enter the command quit.

The interpreter recognizes many options. An option may appear anywhere in the command line, and applies to all files named after it on the line. Many of them include “=” followed by a parameter. The most important are described here. For a complete description see the file `Use.htm`, whose location on your system is indicated at the end of the output generated by typing the command `gs -h`, which displays a summary of the available devices on a given installation, as well as a short overview of the syntax.

```
> gs -h
AFPL Ghostscript 8.51 (2005-04-18)
Copyright (C) 2005 artofcode LLC, Benicia, CA. All rights reserved.
Usage: gs [switches] [file1.ps file2.ps ...]
Most frequently used switches: (you can use # in place of =)
-dNOPAUSE           no pause after page   | -q           'quiet', fewer messages
-g<width>x<height>  page size in pixels    | -r<res>      pixels/inch resolution
-sDEVICE=<devname>  select device         | -dBATCH      exit after last file
-sOutputFile=<file> select output file: - for stdout, |command for pipe,
                                     embed %d or %ld for page #

Input formats: PostScript PostScriptLevel1 PostScriptLevel2 PostScriptLevel3 PDF
Default output device: x11
Available devices:
bbox bit bitcmyk bitrgb bj10e bj200 bjc600 bjc800 bmp16 bmp16m bmp256
bmp32b bmpgray bmpmono bmpsep1 bmpsep8 cdeskjet cdj550 cdjcolor cdjmono
cljet5 cljet5c deskjet devicen djet500 epswrite faxg3 faxg32d faxg4 ijs
jpeg jpeggray laserjet lj5gray lj5mono ljet2p ljet3 ljet3d ljet4 ljet4d
ljetplus nullpage pbm pbmraw pcx16 pcx24b pcx256 pcxcmyk pcxgray pcxmono
pdfwrite pgm pgmraw pgnm pgnmraw pj pjxl pjxl300 pkm pkmraw pksm pksmraw
png16 png16m png256 pngalpha pnggray pngmono pnm pnmraw ppm ppmraw
```

<sup>1</sup>It is beyond the scope of this book to cover how Ghostscript is obtained, installed, possibly compiled, and set up; Thomas Merz' excellent book on PostScript and PDF [14] has a detailed appendix that should answer most questions; the (now somewhat outdated) Ghostscript chapter is freely available at the URL <http://www.cs.wisc.edu/~ghost/doc/merz.htm>.

```

psdcmymk psdrgb psgray psmono psrgb pswrite pxcolor pxlmono spotcmymk
tiff12nc tiff24nc tiff32nc tiffcrle tiffg3 tiffg32d tiffg4 tiffgray
tiff1zw tiffpack tiffsep uniprint x11 x11alpha x11cmymk x11gray2 x11gray4
x11mono xcf
Search path:
. : /usr/local/share/ghostscript/8.51/lib :
  /usr/local/share/ghostscript/8.51/Resource :
  /usr/local/share/ghostscript/fonts
For more information, see /usr/local/share/ghostscript/8.51/doc/Use.htm.
Report bugs to bug-gs@ghostscript.com, using the form in Bug-form.htm.

```

Note the paths for resources, fonts, and documentation printed at the end of the output. This will help you locate the files referenced in this section.

### 22.3.1 Ghostscript options and initialization

The Ghostscript command-line options perform three functions: they let you supply files to run, change options, and give values to various PostScript objects that can be accessed by the program or your files.

The general structure of the options is as follows:

- @*filename*** Read file *filename* and treat its contents as if they were typed on the command line. The options or file names in the file can be on several lines.
- filename arg1* ... and **-+*filename arg1* ...**  
Interpret *filename* as normal but take all remaining arguments and write them in the PostScript userdict as an array of strings called ARGUMENTS.**
- @*filename arg1* ...** As previous, but expand @*filename* arguments.
- c *token* ... and -c *string* ...**  
Interpret the arguments, up to the next argument that begins with - followed by a non-digit or with @, as PostScript code. You can use this, for instance, to force execution of a PostScript `showpage` at the end of a file `picture.ps` by the command `gs picture.ps -c showpage`. More commonly, command lines are ended with `-c quit` to make Ghostscript quit after the files are finished.
- D*name=value* and -d*name=value***  
Define a name in systemdict with the given value, which must be a valid PostScript token. If no value is supplied, it is set to `true`.
- f*filename*** Run the file, even if its name begins with a - or an @. -f by itself does nothing, but it can be used to end a set of -c options.
- g*n1xn2*** Specifies the width and height of the output device in pixels (equivalent to the two options `-dDEVICEWIDTH=n1` and `-dDEVICEHEIGHT=n2`).
- I*directories*** Add the list of directories to the front of the search path for system files; directories are separated by ":" (Linux) or ";" (Microsoft Windows).
- P** Look first in current directory for system files (this is the default).
- P-** Do *not* look first in current directory for system files.
- q** Run quietly, with no startup messages.
- r*n1xn2* and -r*nb*** (equivalent to **-r*nbxn2***)



Sets the resolution of the output device, e.g., a printer with varying resolution, or a bitmap file. The second simpler alternative is the more common one, whereas the more generic first form allows you to handle printing devices that have different  $x$  and  $y$  resolutions. It has the same effect as using `-dDEVICEXRESOLUTION= $n1$`  and `-dDEVICEYRESOLUTION= $n2$` .

`-S $name$ =string` and `-s $name$ =string`

Defines a name in `systemdict` with the given string as value.<sup>1</sup>

`-u $name$`       Undefine a name, cancelling `-d` or `-s`.

`-`      Tell Ghostscript that the standard input is coming from a file or a pipe. Ghostscript reads from `stdin` until reaching an end-of-file, executing it like any other file, and then continues processing the command line. At the end of the command line, Ghostscript exits, rather than going back to an interactive prompt.

### 22.3.1.1 Parameter switches `-d` and `-s`

The parameter switches `-d` and `-s` define initial values for PostScript names. They can be parameter names that control the interpreter or the graphics engine, but they can also define a value for any device parameter of the initial device (defined with `-sDEVICE=`). In the following we describe the more important parameters. A complete list can be found in the documentation file `Use.html`.

#### Rendering parameters

`-dCOLORSCREEN`, `-dCOLORSCREEN=0`, `-dCOLORSCREEN=false`

Force the use of separate halftone screens on devices with a resolution over 150 dpi; `-dCOLORSCREEN=0` uses separate screens with the same frequency and angle; `-dCOLORSCREEN=false` forces the use of a single binary screen. The default behavior is to use separate screens with different angles if the device has fewer than 5 bits per color, and a single binary screen otherwise.

`-dDITHERPPI=1pi`      Forces all devices to be considered high-resolution, and forces use of a halftone screen or screens with *1pi* lines per inch, disregarding the actual device resolution.

`-dDOINTERPOLATE` and `-dNOINTERPOLATE`

The first option turns on image interpolation for all images, improving image quality for scaled images at the expense of speed, while the second, turns interpolation off. `-dNOINTERPOLATE` overrides `-dDOINTERPOLATE` if both are specified.

`-dTextAlphaBits= $n$`  and `-dGraphicsAlphaBits= $n$`

These options should be used to guarantee high quality rasterizations by controlling the use of subsample antialiasing for text and graphics, which are enabled separately. Ideally, the subsampling box size  $n$  should be 4 for optimum output, but smaller values (1 or 2) can be used for faster rendering. In particular when strange lines are encountered within solid areas, try rendering that file again with `-dGraphicsAlphaBits=1`.

`-dAlignToPixels= $n$`       Chooses glyph alignment to integral pixel boundaries (value 1) or to subpixels (value 0, the default). The latter corresponds to the use internally of a smaller raster grid for text antialiasing. Setting `-dAlignToPixels=0` can improve rendering of poorly hinted fonts, but may impair the appearance of well-hinted fonts.

<sup>1</sup>The difference between this and `-d` is that the value is a PostScript *string*, not a token. For instance, the option `"-d myvar=123"` writes `"/myvar 123 def"`, whereas `"-s myvar=123"` writes `"/myvar (123) def"`.

- d**UseCIEColor** Sets `UseCIEColor` in the page device dictionary, remapping device-dependent color values through a CIE color space. This can improve conversion of CMYK documents to RGB.
- d**NOSUBSTDEVICECOLORS** Prevents the substitution of the `ColorSpace` resources (`DefaultGray`, `DefaultRGB`, and `DefaultCMYK`) for the `DeviceGray`, `DeviceRGB`, and `DeviceCMYK` color spaces. This switch is primarily useful for PDF creation using the `pdfwrite` device when retaining the color spaces from the original document is important.

### Page parameters

- d**FIXEDMEDIA** Fix the media size after initialization, forcing pages of other sizes or orientations to be clipped. Implied by the `-g` option.
- d**FIXEDRESOLUTION** Fix media resolution after initialization. Implied by the `-r` option.
- d**DEVICEWIDTHPOINTS**=*w* and -d**DEVICEHEIGHTPOINTS**=*h*  
Set the initial page width and height to *w* and *h*, respectively (units are PostScript points).
- s**DEFAULTPAPERSIZE**=*a4* Replace the device default paper size with *a4* (see Table 22.1 on page 124 for a list of known paper sizes).

### Font-related parameters

- d**NOFONTMAP** Suppress the normal loading of the `Fontmap` file.
- d**NOFONTPATH** Do not consult the `GS_FONTPATH` environment variable.
- d**NOPLATFONTS** Disable the use of fonts supplied by the underlying platform (X Windows or Microsoft Windows).
- s**FONTMAP**=*fn1;fn2;...* Specify one or more alternate names for the `Fontmap` file (on Linux the separator between the filenames is “:”).
- s**FONTPATH**=*dir1;dir2;...* Specify a list of directories to be scanned when looking for fonts not found on the search path (on Linux the separator between the filenames is “:”). Overrides the environment variable `GS_FONTPATH`.
- s**SUBSTFONT**=*fontname* Substitute *fontname* for all unknown fonts; Ghostscript normally tries to find a suitable font, and uses `Courier` as a last resort.

### Resource-related parameters

- s**GenericResourceDir**=*path* Specify path to generic resource files (Default `./Resource/` on Linux, and an equivalent one on other platforms).
- s**FontResourceDir**=*path* Specify path to font resource files (Default `./Font/` on Linux, and an equivalent one on other platforms).

### Interaction-related parameters

- d**BATCH** Exit after processing all files named on the command line (equivalent to ending the command line with `-c quit`).

- dNOPAGEPROMPT** Disable prompt, but not pause, at the end of each page.
- dNOPAUSE** Disable prompt and pause at the end of each page. Normally this is used along with -dBATCH to direct output to a printer or a file.
- dNOPROMPT** Disable prompt when expecting interactive input, as well as the end-of-page prompt (-dNOPAGEPROMPT). Useful for piping input directly into Ghostscript.
- dQUIET** Suppress messages on standard output.
- sstdout=filename** Redirect PostScript %stdout to a file or stderr, to avoid it being mixed with device stdout, i.e., -sstdout=%stderr redirects stdout to stderr, while -sstdout=%stdout or -sstdout=- cancel redirection.
- dTTYPAUSE** Read a character from /dev/tty, rather than standard input, at the end of each page. Useful if input comes from a pipe (-dTTYPAUSE overrides -dNOPAUSE).

### *Device and output selection parameters*

- dNODISPLAY** Initialize Ghostscript with a null device (a device that discards the output image) rather than the default device or the device selected with -sDEVICE=. Useful when running PostScript code whose purpose is to compute rather than to display, e.g., when converting PostScript to PDF or calculating a bounding box (see Section 22.3.5).
- sDEVICE=device** Define an alternate initial output device (see Section 22.3.3).
- sOutputFile=filename** Select an alternate output file (or pipe) for the initial output device, as described above.

### *EPS parameters*

- dEPSCrop** Crop an EPS file to the bounding box. Useful when converting an EPS file to a bitmap.
- dEPSFitPage** Resize an EPS file to fit the page. Useful for enlarging an EPS file to fit the paper size when printing.

### *Other parameters*

- dDOPDFMARKS** Use pdfmark for bookmarks, annotations, links and cropbox when processing PDF files.
- dSAFER** Stop the PostScript deletefile and renamefile operators from working, disallow the ability to use piped commands, and do not allow files to be opened for writing (apart from %stdout and %stderr). It is sensible to set this when interpreting documents which do not originate with a trusted party.

When Ghostscript starts, it needs to find a set of initialization files (much of the interpreter's functions are written in PostScript itself and loaded when the program starts). Normally, the program is set up to look in the current directory for your files and in a system directory for the program files, but you may need to override this. You can ask for extra directories to be searched by using the -I option on the command line; or you can set an environment variable `GS_LIB` to point to directories. Directories can be specified as a list, separated by ":" (Linux) or ";" (Microsoft Windows).

### 22.3.2 Ghostscript and fonts

In addition to its system files, Ghostscript needs to find fonts. Unlike most PostScript printers, it normally has no built-in fonts at all, but loads everything as PostScript Type 1 or TrueType fonts from your disk. Ghostscript comes with a basic set of 35 fonts to match those found in most PostScript printers. These are good-quality Type 1 PostScript fonts made freely available by the German font company URW. It starts by looking for a file catalog called `Fontmap` (you can change the name with `-sFONTMAP`) among the system files. This file specifies how to relate PostScript font names to actual files on the system. It then searches any of the directories specified with the `GS_FONTPATH` environment variable or `-sFONTPATH` option for files that appear to contain PostScript fonts. If it decides that they are valid fonts, Ghostscript adds those files and fonts to its internal copy of the `Fontmap` catalog.

When your files are read, font requests are satisfied using the `Fontmap` catalog; if no match is found, the default font is used. If on your system other PostScript Type 1 fonts are available in one or more directories their path should be added to `GS_FONTPATH`, so that all fonts are automatically available. Note that you should also add the names of the fonts and the names of the files where they reside into the font catalog, which is described next.

The `Fontmap` catalog file consists of a set of lines describing a font; each line has the name of a PostScript font (prefixed with `/`), followed by some white space, and then the name of a file on disk, in parentheses. Alternatively, the second part of the line can be the name of an already defined font, to act as an alias. The line must be terminated by a semicolon. Comments are added by starting a line with a `%` character.

As an example, the familiar Palatino family is provided by URW's Palladio fonts; this extract from `Fontmap` shows how the names are mapped to actual files on disk:

```
/URWPalladioL-Roma      (p0520031.pfb) ;
/URWPalladioL-Ital      (p0520231.pfb) ;
/URWPalladioL-Bold      (p0520041.pfb) ;
/URWPalladioL-BoldItal  (p0520241.pfb) ;
```

The full path name of the file is not normally given, since the fonts are expected to be found in the font directory path. In order to allow files to call up the font by the real Palatino names, a set of aliases is provided:

```
/Palatino-Roman         /URWPalladioL-Roma ;
/Palatino-Italic        /URWPalladioL-Ital ;
/Palatino-Bold          /URWPalladioL-Bold ;
/Palatino-BoldItalic    /URWPalladioL-BoldItal ;
```

Since Ghostscript normally searches the directories for suitable font files anyway, the first example above is actually redundant, but the alias entries in `Fontmap` are vital, unless you have a copy of real Palatino on your disk. Users of Adobe Type Manager *do* have real Adobe fonts that Ghostscript can use, and special versions of the `Fontmap` file are provided for this and other setups.

### 22.3.3 Selecting an output device

Ghostscript saves or displays its results in a particular format on an “output device”, of which it has a large variety allowing it to support vector and raster file output, screen display, and many native printer formats. The command `gs -h` lists all devices available with your version. From inside Ghostscript you

can also find out what devices are available by typing the following at the interactive prompt.

```
GS> devicenames ==
[/bjc600 /bit /cdeskjet /tiffg4 /deskjet /pnm /cljet5 /x11gray2
/pcxgray /png256 /pjxl300 /tiff32nc /ljet3d /pksm /psdrgb /bmpsep8
/pbmraw /pswrite /bjc800 /bit rgb /cdjcolor /tiff1zw /djet500 /pnmraw
/cljet5c /x11gray4 /pcx16 /png16 /unipri nt /tiffsep /ljet4 /pksmraw
/nullpage /bmp16 /pgm /epswrite /faxg3 /bitcmyk /cdj mono /tiffpack
/laserjet /ppm /spotcmyk /x11mono /pcx256 /pngalpha /ijs /psmono
/ljet4d /tiffcrle /bmp256 /pgmraw /pxlmono /x11 /faxg32d /png16m
/cdj550 /tiff12 nc /ljetplus /ppmraw /devicen /bmpmono /pcx24b /jpeg
/bj10e /psgray /lj5mono /ti ffg3 /bmp16m /pgnm /pxlcolor /x11alpha
/faxg4 /pnggray /pj /tiff24nc /ljet2p /pk m /xcf /bmpgray /pcxcmyk
/jpeggray /bj200 /psrgb /lj5gray /tiffg32d /bmp32b /pgn mraw /bbox
/x11cmyk /pcxmono /pngmono /pjxl /tiffgray /ljet3 /pkmraw /psdcmyk
/bmpsep1 /pbm /pdfwrite]
```

A complete description of all devices supported by Ghostscript and their options can be found in the file `Devices.html` in the directory containing the documentation.

As output device Ghostscript uses the one specified with the command line option `-sDEVICE=device` or the default device (often the display) otherwise. This switch must precede the name of the first input file, and only its first use has any effect. An example is:

```
> gs -sDEVICE=deskjet myfile.ps
```

The output device can also be set through the `GS_DEVICE` environment variable, or from inside Ghostscript on the interactive prompt as follows:

```
GS> (deskjet) selectdevice
```

All output then goes to the deskjet printer instead of the display until you change the device, e.g.,

```
GS> (laserjet) selectdevice
GS> (myfile.ps) run
GS> (deskjet) selectdevice
```

Some printers can manage different resolutions, often trading resolution against printing speed. The resolution is selected with the `-r` switch (see Section 22.3.1). This option is also useful for controlling the density of pixels when rasterizing to an image file.

The output of Ghostscript can be sent to a file with the `-sOutputFile=` switch, e.g.,

```
> gs -sOutputFile=myfile.xyz
```

This file will then have to be sent to the printer via the appropriate procedure on your operating system. The special filename “-” tells Ghostscript to send its output to standard output (the command shell).

You can instruct Ghostscript to put each page of output in a series of similarly named files. This is achieved by placing a template “%d” in the filename which Ghostscript will replace with the page number, where the number of digits in the filename can also be controlled, e.g.,

```
gs -sOutputFile=fil-%d.png produces ABC-1.png, ..., ABC-10.png
gs -sOutputFile=fil-%03d.png produces ABC-1.png, ..., ABC-999.png
gs -sOutputFile=fil-%04d.png produces ABC-1.png, ..., ABC-9999.png
```

Table 22.1: Standard US, ISO, and Japanese paper sizes

U.S. standard paper sizes											
Name	inches		mm		points		Name	inches		mm	
	$W \times H$	$W \times H$	$W \times H$	$W \times H$	$W \times H$	$W \times H$		$W \times H$	$W \times H$	$W \times H$	$W \times H$
11x17	11.0	17.0	279	432	792	1224	archE	36.0	48.0	914	1219
ledger	17.0	11.0	432	279	1224	792	archD	24.0	36.0	610	914
legal	8.5	14.0	216	356	612	1008	archC	18.0	24.0	457	610
letter	8.5	11.0	216	279	612	792	archB	12.0	18.0	305	457
lettersmall	8.5	11.0	216	279	612	792	archA	9.0	12.0	229	305
ISO standard paper sizes											
a0	33.1	46.8	841	1189	2384	3370	a5	5.8	8.3	148	210
a1	23.4	33.1	594	841	1684	2384	a6	4.1	5.8	105	148
a2	16.5	23.4	420	594	1191	1684	a7	2.9	4.1	74	105
a3	11.7	16.5	297	420	842	1191	a8	2.1	2.9	52	74
a4	8.3	11.7	210	297	595	842	a9	1.5	2.1	37	52
a4small	8.3	11.7	210	297	595	842	a10	1.0	1.5	26	37
isob0	39.4	55.7	1000	1414	2835	4008	c0	36.1	51.1	917	1297
isob1	27.8	39.4	707	1000	2004	2835	c1	25.5	36.1	648	917
isob2	19.7	27.8	500	707	1417	2004	c2	18.0	25.5	458	648
isob3	13.9	19.7	353	500	1001	1417	c3	12.8	18.0	324	458
isob4	9.8	13.9	250	353	709	1001	c4	9.0	12.8	229	324
isob5	6.9	9.8	176	250	499	709	c5	6.4	9.0	162	229
isob6	4.9	6.9	125	176	354	499	c6	4.5	6.4	114	162
JIS standard paper sizes											
jisb0			1030	1456			jisb4			257	364
jisb1			728	1030			jisb5			182	257
jisb2			515	728			jisb6			128	182
jisb3			364	515							

For normal documents, “%03d” is usually all right.<sup>1</sup> This function does not work with document-directed devices, such as `pdfwrite` and `pswrite`. In the case of PDF generation one of more pages can be extracted with the `-dFirstPage` and `-dLastPage` switches (see Section 22.3.5).

### 22.3.3.1 Measurements

PostScript and Ghostscript use a certain number of measures: inches (in), centimeters (cm), millimeters (mm), and points (pt). One has 1 in = 2.54 cm = 72 pt where the latter are so called “big points,” or PostScript points, so that 1 cm  $\approx$  25.35 pt. Table 22.1 provides a list of paper sizes expressed in these units. Another common unit is “bits per pixel” (bpp), also known as the “bit depth” or the “pixel depth”, and which corresponds to the number of digital bits used to represent the color of each pixel.

<sup>1</sup>The character “%” has a special meaning on Microsoft Windows, hence you must double it “%%”, e.g.,  
`gs -sOutputFile=file-%%03d.png.`

### 22.3.3.2 Paper sizes

By default Ghostscript uses U.S. letter paper as its default page size. You can change this and set it to one of the internationally recognized dimensions of Table 22.1 on the preceding page, as follows:

```
> gs -sPAPERSIZE=a4 ...
```

More generally, the page width (*w*) and height (*h*) can be specified in points by using a pair of switches, as follows:

```
> gs -dDEVICEWIDTHPOINTS=w -dDEVICEHEIGHTPOINTS=h ...
```

Documents often specify a paper size which takes precedence over the default size. If this is not desired (e.g., one wants to print a “letter” sized document on an “a4” printer) a specific paper size can be forced by selecting the desired size and add the `FIXEDMEDIA` switch, as follows:

```
> gs -sPAPERSIZE=a4 -dFIXEDMEDIA ...
```

The installed default paper size on your installation can be changed by editing the initialization file `gs_init.ps`, which usually resides in a Ghostscript’s `lib` directory (see the output of the command `gs -h` for its whereabouts). In that file you should locate the line

```
% /DEFAULTPAPERSIZE (a4) def
```

and uncomment it if you want “a4” as default, as follows:

```
/DEFAULTPAPERSIZE (a4) def
```

Instead of “a4” you can set any of the known paper sizes of Table 22.1 on the facing page.

### 22.3.3.3 Using pipes

The input and output files can, on most operating systems, be replaced by pipes, in which case you should use “-” as placeholder for the filename. This, however, only works for PostScript sources, since PDF files have a random access structure and thus cannot be read from standard input or used in pipes. Two examples follow.

```
> ... | gs [options] -
> gs [options] myfile.ps -q -sOutputFile=- | lpr
```

In the first line we pipe data into `gs`, while in the second line we read a file and pipe output into the `lpr` command (the `-q` switch is needed to prevent Ghostscript from writing messages to standard output which become mixed with the intended output stream.)

### 22.3.3.4 Preparing image files

When preparing image files by rasterizing one often uses switches to specify the output file (`-sOutputFile`), define the resolution (`-r`, default 72 dpi), set antialiasing characteristics (`-dTextAlphaBits` and `-dGraphicsAlphaBits`), and to suppress interactive prompts and enable some security checks on the file to be run (`-dSAFER`, `-dBATCH`, and `-dNOPAUSE`).

## PNG

Portable Network Graphics (PNG, see <http://www.libpng.org/pub/png/pngintro.html>) is the recommended format for high-quality bitmap images. PNG features full quality color, transparency, excellent lossless compression, and is widely supported.

Possible PNG devices supported by Ghostscript are: for normal use `png16m` for 24-bit RGB color, or `pnggray` for grayscale are recommended devices; for special needs `png256`, `png16`, and `pngmono` provide, respectively, 8-bit color, 4-bit color and black-and-white support. Moreover, `pngalpha` gives you 32-bit RGBA color with transparency indicating pixel coverage. This device lets you specify the background color in the RGB model with the option `-dBackgroundColor=16#RRGGBB` (default white = 16#ffffff).

Examples of how Ghostscript can convert PostScript or PDF to PNG are:

```
> gs -dSAFER -dBATCH -dNOPAUSE -sDEVICE=png16m -dGraphicsAlphaBits=4 \
    -sOutputFile=myfile.png myfile.ps
> gs -dSAFER -dBATCH -dNOPAUSE -r150 -sDEVICE=pnggray -dTextAlphaBits=4 \
    -sOutputFile=myfile-%02d.png myfile.pdf
```

## JPEG

Joint Photographic Experts Group (JPEG, see <http://www.jpeg.org/>) images are specifically intended for continuous-tone images such as photographs, not for the usual kind of graphics images that are produced with PostScript. In fact, for anything other than very simple drawings JPEG's lossy compression will result in poor quality output regardless of the input.<sup>1</sup> The `jpeg` and `jpeggray` devices generate for color, respectively, grayscale JPEG images, eg

```
> gs -dSAFER -dBATCH -dNOPAUSE -sDEVICE=jpeg -sOutputFile=myfile.jpg myfile.ps
```

Several options are available to control the JPEG “quality settings” (see the documentation for more details).

## PNM

The “portable network map” (PNM) family of formats are very simple uncompressed image formats commonly used on Unix-like systems. The PNM format is an abstraction of the lowest common denominator file formats for color (PBM), grayscale (PGM), and black-and-white (PPM).<sup>2</sup> Ghostscript supports a wide variety of such data formats, e.g., `pbm`, `pbmraw`, `pgm`, `pgmraw`, `ppm`, `pnm`, `pkm`, and many others.

## TIFF

Tagged Image File Format (TIFF, see <http://en.wikipedia.org/wiki/TIFF>) is a file format mainly used for storing images — especially high color-depth ones —, including photographs and line art.<sup>3</sup> Although nowadays largely superseded by PNG, TIFF is still widely supported by image-manipulation desktop and page layout tools, and especially useful in connection with scanners, optical character recognition tools, fax machines, etc.

<sup>1</sup>The Web page <http://www.faqs.org/faqs/jpeg-faq/> explains some of the issues when choosing an output format.

<sup>2</sup>The `netpbm` Project (<http://netpbm.sourceforge.net/doc/index.html>) has over 200 programs to transform between these and other graphics formats. The `ImageMagick` Project (<http://www.imagemagick.org>) also supports over 90 different formats and allows conversions between them.

<sup>3</sup>The specification is at <http://partners.adobe.com/public/developer/tiff/index.html>.



Five color TIFF drivers produce uncompressed output: `tiffgray` (8-bit gray), `tiff12nc` (12-bit RGB, 4 bits per component), `tiff24nc` (24-bit RGB output, 8 bits per component), `tiff32nc` (32-bit CMYK output, 8 bits per component). Moreover, `tiffsep` creates multiple output files, a single 32 bit composite CMYK file (`tiff32nc` format) and several `tiffgray` files, one for each separation (see the documentation for details of how to specify separations).

Other TIFF drivers produce black-and-white output with different compression modes: `tiffcrle` and `tiffg3` (G3 fax encoding without and with end-of-lines), `tiffg32d` (2-D G3 fax encoding), `tiffg4` (G4 fax encoding) `tiff1zw` and `tiffpack` (LZW-compatible and PackBits compression).

Ghostscript supports a variety of fax encodings, either encapsulated in TIFF, as described above and as raw files (`faxg3`, `faxg32d` and `faxg4`).

### Microsoft Windows bitmap formats

BMP (or DIB for device-independent bitmap) is a simple, uncompressed, bitmapped graphics format developed by Microsoft and IBM. Images can have a color depth up to 24 bits (16.7 million colors). 8-bit images can also be greyscale instead of indexed color. An alpha channel (for transparency) may be stored in a separate file, or can be integrated in a 32-bit version that has been introduced with Windows XP, but is not yet generally supported in image software. In Ghostscript BMP is supported by the devices `bmpmono`, `bmpgray`, `bmpsep1`, `bmpsep8`, `bmp16`, `bmp256`, `bmp16m`, `bmp32b`.

PCX (<http://www.qzx.com/pc-gpe/pcx.txt>) is an image file format that uses a simple form of run-length encoding (a lossless compression algorithm). Although presently largely replaced by formats with better compression, such as JPEG and PNG, PCX is still often used on Microsoft Windows. PCX files use a color palette with alternate color spaces and so can be a useful way to output CMYK. In Ghostscript PCX is supported by the devices `pcxmono`, `pcxgray`, `pcx16`, `pcx256`, `pcx24b`, `pcxcmk`.

#### 22.3.3.5 Output for Inkjet and other raster devices

IJS is a protocol for transmission of raster page images. It is a relatively new initiative to improve the quality and ease of use of inkjet printing with Ghostscript. With IJS you can add new drivers, or upgrade existing ones, without recompiling Ghostscript. All driver authors are adapting their drivers for IJS.<sup>1</sup> An example of a command line for an IJS device is:

```
gs -dSAFER -sDEVICE=ijs -sIjsServer=hpijs \
-sDeviceManufacturer=HEWLETT-PACKARD -sDeviceModel='DESKJET 990' \
-dIjsUseOutputFD -sOutputFile=/dev/usb/lp1 -dNOPAUSE -- myfile.eps
```

The `-sIjsServer` switch specifies the pathname for the IJS printer driver. Ghostscript will spawn a new process for this driver and will communicate with it using the IJS protocol. The switches `-sDeviceManufacturer` and `-sDeviceModel` are provided by the manufacturer (values containing spaces should be quoted, as shown in the example). The `-dIjsUseOutputFD` flag indicates that Ghostscript should open the output file and pass a file descriptor to the server. If not set, Ghostscript simply passes the filename set in `-sOutputFile` to the server.

Other relevant Ghostscript parameters are:

**-dBitsPerSample=*n*** Number of bits per sample (default value is 8). For monochrome images, use `-dBitsPerSample=1`.

<sup>1</sup>The IJS web page (<http://www.linuxprinting.org/ijs/>) has more information about IJS, including a listing of IJS-compatible drivers.

- rnb** Resolution (see Section 22.3.1). If the resolution is not specified, Ghostscript queries the IJS server to determine the preferred resolution, otherwise it overrides the value (if any) preferred by the IJS server by the one specified on the command line.
- dDuplex** Enable duplex (two-sided) printing.
- dTumble** Controls the orientation. When Tumble is false, the pages are oriented suitably at the left or right. When Tumble is true, the pages are oriented suitably for binding at the top or bottom.
- sProcessColorModel=name** Set the process color model. Possible values include DeviceGray, DeviceRGB, and DeviceCMYK.

HP provides official drivers for many of their Deskjet printer models. In order to use these drivers, you will need the HP Inkjet Server available from the HP Driver Project (<http://hplip.sourceforge.net/>). Currently, they provide support for nearly 1000 printing models, including color Deskjet and Business Inkjet and color Inkjet Photo printers (Photosmart), color LaserJet Printers and Multi-Function Printers (MFPs), Color Inkjet all-in-ones (Officejet and PSC), monochrome (B&W) LaserJet printers and MFPs (see the web page mentioned for a complete list). Recent versions of the hpijs drivers support the IJS protocol.

The *Gimp-Print* Project (<http://gimp-print.sourceforge.net/>) also provides a large collection of printer drivers with an IJS interface.

### 22.3.3.6 High-level devices

These devices preserve as much as possible the vector drawing elements of the input file.

The `pdfwrite` device outputs PDF (see below or the file `Ps2pdf.htm` in the Ghostscript distribution for details about the many supported options).

The `pswrite` and `epswrite` devices output PostScript and Encapsulated PostScript, respectively. Both these devices have an option `-dLanguageLevel` which can be set equal to the value 1, 1.5, 2 (default), and 3. It sets the PostScript language level for the generated file.

The `pxlmono` and `pxlcolor` devices output HP PCL-XL, a graphic language understood by many recent laser printers.

### 22.3.3.7 Display devices

For preparing screen displays of PostScript or PDF documents Ghostscript is a good choice. A client or viewer can call the Ghostscript engine to do the rasterization and handle the display of the resulting image itself, or Ghostscript can be invoked to handle the display of the image on screen itself by specifying an adequate output device.

On Unix, the most commonly used display device is based on the X Window System. Various possibilities exist.

- x11** the default device for handling the display on X11R6;
- x11alpha** the same as the above but with antialiasing;
- x11mono** black-and-white device for 1-bit monochrome displays;
- x11gray2** two-bit (4-level) monochrome displays;
- x11gray4** four-bit (16-level) monochrome displays.

On Microsoft Windows and with the `gtk+` versions of Ghostscript one can use the `display` device. The display format is set with the `-dDisplayFormat` option, an integer representing a bit string (see the documentation for details). Examples are 16#30804 (Windows RGB), 16#804 (`gtk+` RGB), 16#20101 (Windows monochrome), 16#102 (`gtk+` monochrome), 16#20802 (grayscale), 16#20808 (CMYK), and 16#a0800 (separations).

The option `-dDisplayResolution`, which initializes the resolution for a display device, lets Windows clients set this resolution to the Windows display logical resolution. This can be overridden by the command line option `-rDPI`.

Finally, a special `bbox` “device” just prints the bounding box of each page. It can be run as follows:

```
gs -dSAFER -dNOPAUSE -dBATCH -sDEVICE=bbox frag2.eps
AFPL Ghostscript 8.51 (2005-04-18)
...A few lines deleted
%%BoundingBox: 148 624 228 668
%%HiResBoundingBox: 148.667990 624.467934 227.507987 667.307977
```

The bounding box information is output to `stderr`. Beware that white objects are invisible to this procedure.

### 22.3.4 Interactive Ghostscript versions

Plain `ghostscript` provides no easy facilities to navigate a document, zoom in on parts of the page, rotate pages, print selected pages, etc. Such functions are provided by front ends that offer a convenient interface to `ghostscript`, such as the freely available programs `GSview` for MS Windows developed by Russell Lang, `Mac GS Viewer` for the Macintosh, and `ghostview` for Unix X Windows System. Figure 22.3 shows a PostScript page viewed with Timothy Theisen’s `ghostview`.<sup>1</sup> A portion of the page has been magnified. The menu on the left indicates the functions available, including loading new files, printing selected pages, jumping to different page numbers, and changing magnification, paper size, and orientation.

Figure 22.4 on the next page displays a PDF incarnation of the same page as in Figure 22.3 displayed by `Evince`.<sup>2</sup> The *About* information and the contents of the *Edit* menu are shown.

### 22.3.5 Ghostscript applications

Many applications for handling PostScript programs depend on `Ghostscript` to handle (convert, transform, rasterize) the data, and we look at a few of them here.

These small applications are sometimes written as scripts that call PostScript programs; these are usually provided as both Unix shell scripts and MS-DOS batch files and they often simply give the command-line options needed for `Ghostscript`. Hence, users of other operating systems can easily adapt these files to their needs.

#### 22.3.5.1 Extracting text from PostScript files

`Ghostscript` comes with a PostScript file called `ps2ascii.ps`, which extracts the ASCII text from a PostScript file; nothing is displayed, but the text is written to standard output. It is wrapped up in a script called `ps2ascii`, used as follows:

```
> ps2ascii myfile.ps [myfile.txt]
```

<sup>1</sup>`ghostview`, the first X11 interface to `Ghostscript`, gave life to several derived programs (see <http://www.cs.wisc.edu/~ghost/gv/> for a list), such as Johannes Plass’s `gv` on Linux. Currently, Jaka Mociak is developing a Gnome-based interface `ggv` (<http://directory.fsf.org/print/misc/ggv.html>).

<sup>2</sup>`Evince` (<http://www.gnome.org/projects/evince/>) is a document viewer for multiple document formats. It currently supports PDF, PostScript, DJVU (a new compression technology, see <http://www.djvuzone.org/>), TIFF, and DVI. The goal of `evince` is to replace the multiple document viewers that exist on the Gnome Desktop with a single simple application.

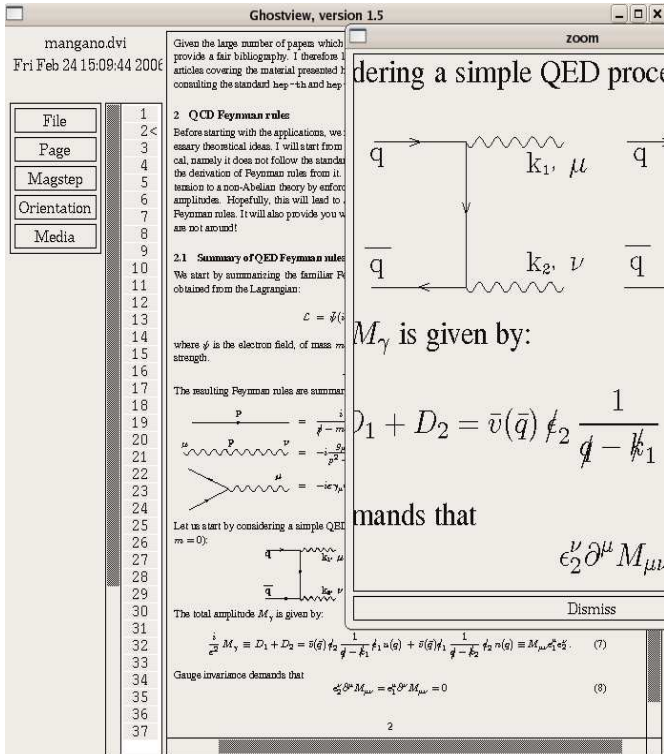


Figure 22.3: Example of the use of ghostview.

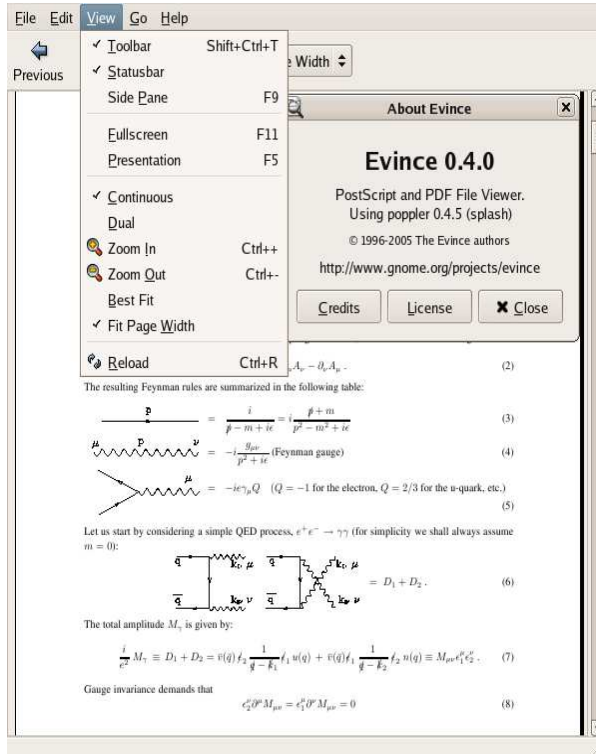


Figure 22.4: Example of the use of evince.

Another utility, pstotext, (<http://www.cs.wisc.edu/~ghost/doc/pstotext.htm>) originally written by Andrew Birrell and Paul McJones, is a little more robust. Like `ps2ascii.ps`, it reads one or more PostScript files and writes out a representation of the plain text that would be displayed if the PostScript file were printed. Internally pstotext uses Ghostscript and loads a PostScript library that locates each string in the file, and enough additional information to approximate the string's bounding rectangle. pstotext then post-processes this information and outputs a sequence of words delimited by space, newline, and formfeed. In particular, eight-bit input codes are translated into an ISO 8859-1 character code, when available, or into a sequence of characters otherwise, e.g., “---” for em dash.

### 22.3.5.2 Creating EPS Interchange files

The utility script `ps2epsi` takes a PostScript file as input and generates an output file that conforms to Adobe's Encapsulated PostScript Interchange (EPSI) format.<sup>1</sup> This special form of Encapsulated PostScript adds a monochrome bitmap version of the final displayed page (in the form of PostScript comments) to the beginning of the file. This is useful for text processors or desktop publishing tools like FrameMaker which use it to display an approximate representation of the picture in the regular display. The script is used as follows:

```
> ps2epsi infile [outfile]
```

If the output file is not specified then the same name as for the input file name is used, but with the extension `.epsi`.

<sup>1</sup>See [http://partners.adobe.com/public/developer/en/ps/5002.EPSF\\_Spec.pdf](http://partners.adobe.com/public/developer/en/ps/5002.EPSF_Spec.pdf).

Note that not all PostScript files can be encapsulated, because there are restrictions in what is permitted in a PostScript file for it to be properly encapsulated. `ps2epsi` does its best to help encapsulation, it also automatically calculates the bounding box, but in some cases encapsulation necessarily fails because of the nature of the original PostScript file.

### 22.3.5.3 Preparing Adobe Illustrator files

Adobe's Illustrator program is frequently used for preparing and editing sophisticated illustrations. Illustrator uses highly structured and simplified PostScript as its native storage format, and ghostscript comes with a PostScript program, `ps2ai.ps`, that tries to convert normal PostScript into a form Adobe Illustrator can digest. There are some restrictions in the PostScript code that it can understand, but in general the program does quite a good job. The procedure is called as follows:

```
> gs -q -dNODISPLAY ps2ai.ps infile.ps > outfile.ai
```

Inside the file `ps2ai.ps` a few PostScript logical switches are available for finetuning the process (e.g., `joutln` for replacing fonts by their outlines, see that file for more details.)

### 22.3.5.4 Converting between PostScript and PDF

ghostscript comes with two scripts, `ps2pdf` and `pdf2ps`, that convert, respectively, PostScript to PDF and, conversely, PDF to PostScript.

`ps2pdf` is a work-alike for nearly all the functionality of Adobe's Acrobat Distiller program. It is implemented as a very small command script (batch file) that invokes Ghostscript, selecting a special "output device" called `pdfwrite`. It is used as follows.

```
> ps2pdf infile.ps [outfile]
> pdf2ps infile.pdf [outfile]
```

In fact, there are several instances of `ps2pdf`:<sup>1</sup>

- `ps2pdf12` produces PDF 1.2 output (for Acrobat 3 or later);
- `ps2pdf13` produces PDF 1.3 output (for Acrobat 4 or later);
- `ps2pdf14` produces PDF 1.4 output (for Acrobat 5 or later);
- `ps2pdf` is currently the same as `ps2pdf12`.

It has been found that certain characters (like "fi" ligatures) have incorrect spacing with `ps2pdf12`, whereas `ps2pdf14` generates code that some printers cannot digest. Hence, `ps2pdf13` seems to be the best compromise at the moment.

Ghostscript is normally built to interpret both PostScript and PDF files, examining each file to determine automatically whether its contents are PDF or PostScript. All the normal switches and procedures for interpreting PostScript files also apply to PDF files, with a few exceptions. In addition, the `pdf2ps` utility uses Ghostscript to convert PDF to (Level 2) PostScript.

Here are some command line options specific to PDF:

<b>-dFirstPage=pn</b>	Page number where interpretation of document will start.
<b>-dLastPage=pn</b>	Page number after which interpretation of document will stop.
<b>-dPDFFitPage</b>	Scale the PDF input to fit the current device page size.
<b>-dPrinted</b>	Display file using "printer" option for annotation and images.

<sup>1</sup>All these scripts call the script `ps2pdfwr`, which contains something like the following code:

```
gs -dSAFER -q -dNOPAUSE -dBATCH -sDEVICE=pdfwrite "-sOutputFile=out.pdf" -f "in.ps".
```

**-dPrinted=false** Display file using the “screen” option for annotation and images. When no `-dPrinted` option is present, the output will use the printer options for output devices where the `OutputFile` parameter is present, and the screen options for the other devices.

**-sPDFPassword=password**  
Sets the user or owner password for decoding encrypted PDF files.

The Ghostscript switches `-dparameter=value` or `-sparameter=string` can be used to set Adobe Acrobat Distiller parameters. For instance, by default Ghostscript determines the page orientation for viewing based on the dominant text orientation on the page. Sometimes, when the page has text in several orientations or has no text at all, wrong orientation can be selected. To control this behavior, you can set the Acrobat Distiller parameter `AutoRotatePages`, which controls page orientation, as follows:

```
> ps2pdf13 -dAutoRotatePages=/None in.ps out.pdf
```

Another interesting option is `-dPDFSETTINGS=configuration`, which lets you preset the Acrobat Distiller parameters to one of the following predefined settings:

- `/screen` selects low-resolution output (like Acrobat Distiller’s *Screen Optimized*).
- `/ebook` selects medium-resolution output (like Acrobat Distiller’s *Ebook*).
- `/printer` selects printer output (Like Acrobat Distiller’s *Print Optimized*).
- `/prepress` selects high quality output (like Acrobat Distiller’s *Prepress Optimized*).
- `/default` selects output intended to be useful across a wide variety of uses, possibly at the expense of a larger output file.

See the documentation file `Ps2pdf.htm` distributed with `ghostview` for a complete description of the numerous options available to control PDF generation.

`pdf2ps`, or the more general `ps2ps` script, transform a PDF, or any PostScript or PDF file for the latter into PostScript level 2 constructs. These scripts use the output device “`ps2write`” internally.

```
> ps2ps [options] infile.ps|eps|pdf outfile.ps
> ps2pdf [options] infile.pdf outfile.ps
```

The *options* given on the command line may include any of the Ghostscript switches.

Since `ps2write` makes use of some printer parameters while converting high level objects into PostScript Level 2 objects, these values should be set in accordance with the target printer. In particular, care should be taken with fonts, such as CID fonts, which are PostScript Level 3 objects and which will be converted by `ps2write` into bitmaps at the resolution specified. Additionally, for controlling the conversion process, `ps2ps` and `pdf2ps`’s *options* may include the same `-dparameter=value` or `-sparameter=string` switches as `ps2pdf` for setting Acrobat Distiller parameters (in fact, `ps2write` and `pdfwrite` handle an identical set of *Acrobat Distiller* parameters).

An important option is `-dFitPages=boolean`, which, if set to “`true`” will scale pages down to fit into the real page size. This scaling may result in poor quality rendering, especially for fonts which Ghostscript had converted into bitmaps.

`pdfopt` converts a PDF file into its linearized form, which is organized in an optimal way for enhancing incremental access over the network, and in particular to display individual pages of a file more quickly (see [3, Appendix F]).

```
> pdfopt infile.pdf outfile.pdf
```



dvipdf is another derivative of ghostview. It converts a T<sub>E</sub>X DVI file into PDF by first running dvips in *quiet* mode (`-q` option) piping the output into Ghostscript (writing to the `pdfwrite` device).

```
> dvipdf [Ghostscript options] infile.dvi [outfile.pdf]
```

It is thus a convenient way of combining the generation of PostScript from L<sup>A</sup>T<sub>E</sub>X with dvips and running ps2pdf13 on the output yourself in cases where no special options to dvips need to be specified. More details on how to generate PDF from L<sup>A</sup>T<sub>E</sub>X documents are given in Section 22.5.

### 22.3.5.5 Concatenation of files

Two or more PostScript or PDF files can be concatenated as follows:

```
> gs -q -dNOPAUSE -dBATCH -sDEVICE=pswrite -sOutputFile=out.ps a.ps b.ps c.ps
> gs -q -dNOPAUSE -dBATCH -sDEVICE=pdfwrite -sOutputFile=out.pdf a.pdf b.pdf c.pdf
```

### 22.3.5.6 Page Selection

One can select one or more pages from a PostScript or PDF file and write the to the output file. An example choosing to extract pages 3 to 8 from the input file follows.

```
> gs -q -dNOPAUSE -dBATCH -sDEVICE=pswrite -sOutputFile=out.ps \
    -dFirstPage=3 -dLastPage=8 in.ps
> gs -q -dNOPAUSE -dBATCH -sDEVICE=pdfwrite -sOutputFile=out.pdf \
    -dFirstPage=3 -dLastPage=8 in.pdf
```

### 22.3.5.7 Getting the correct Bounding Box

Russell Lang's epstool can create or extract preview images in EPS files and calculate optimal bounding boxes. Internally epstool delegates most of the work to ghostscript.

Three kinds of preview information about images can be used in EPS files.

**Interchange** Preview data are stored as comments inside the PostScript header in hexadecimal format between a `%%BeginPreview`, `%%EndPreview` pair (used mostly on Linux).

**MS-DOS EPS** The preview is a TIFF or Windows Metafile preceeded by a binary header. This preview part has to be removed before sending the file to a printer (Microsoft Windows).

**PICT** The preview is in PICT format stored in the resource fork of the file (Macintosh).

Many options exist to add or remove these formats of preview data, or to (re)calculate the bounding box. A few examples will show some of the more interesting features.

Add a compressed TIFF 6 preview image. By default, the preview will be full color (24 bits/pixel), but `greyscale` (`--device bmpgray` or `--device pgmraw`), or `monochrome` (`--device bmpmono` or `--device pbmraw`) can also be specified.

```
> epstool --add-tiff6p-preview infile.eps outfile.eps
```

Generate a TIFF preview using ghostscript. In this case we specified a compressed monochrome G3 fax image, but any ghostscript TIFF device can be used, e.g., `tiffg4`, `tiffpack`.

```
> epstool --add-tiff-preview --device tiffg3 infile.eps outfile.eps
```

Extract the TIFF preview section from a MS-DOS EPS file.

```
> epstool --extract-preview --device tiffg3 infile.eps outfile.tif
```

The `--bbox` option uses the `ghostscript` `bbox` device to calculate the bounding box and then a monochrome TIFF 4 preview is added.

```
> epstool --bbox --add-tiff4-preview infile.eps outfile.eps
```

The `--copy` option copies the EPS file, so that, in conjunction with `--bbox` the bounding box can be recalculated and updated in the output file. No preview is added.

```
> epstool --bbox --copy infile.eps outfile.eps
```

Add a user-supplied Windows Metafile (WMF) to an EPS image.

```
> epstool --add-user-preview infile.wmf infile.eps outfile.eps
```

On Mac OS X add a PICT preview to the resource fork of an EPS file. This resource fork can later be accessed from command line tools by appending `/rsrc` to the filename. The existing resources are overwritten.

```
> epstool --add-pict-preview --mac-rsrc infile.eps outfile.eps/rsrc
```

### **pstoedit**

Wolfgang Glunz's `pstoedit` (<http://www.pstoedit.net/pstoedit/>) is a program which uses `ghostscript` to translate PostScript and PDF graphics into other vector formats. Most of its drivers are freeware. Supported formats include: `tgif` (`.obj`), `xfig` (`.fig`), `groff` (`.pic`), PDF, `gnuplot`, `HPGL`, on Microsoft Windows (Extended) Windows meta files (WMF and EMF), `ldraw`, `MetaPost`,  $\text{\TeX}$ 's picture environment, GNU Metafile, `Mathematica`, all formats handled by `ImageMagick` (using that program to do the conversion), `Micromedia Flash` (`.swf`).

Other drivers are available as shareware, and support the following formats: `FrameMaker` interchange format (MIF), `Computer Graphics Metafile` (CGM), `scalable vector graphics format` (SVG), and, on Microsoft Windows, enhanced EMF, and `Rich Text Format` (RTF),

### **epstopdf**

The  $\text{\TeX}$  Live distribution comes with a script `epstopdf` which outputs a PDF file with the correct bounding box from an EPS input file. Its usage is as follows:

```
> epstopdf --help
EPSTOPDF 2.9.3draft, 2003/04/20 - Copyright 1998-2002 ...
Syntax: epstopdf [options] <eps file>
Options:
  --help:                print usage
  --outfile=<file>:      write result to <file>
  --(no)filter:          read standard input    (default: false)
  --(no)gs:              run ghostscript        (default: true)
  --(no)compress:        use compression        (default: true)
  --(no)hires:           scan HiResBoundingBox (default: false)
  --(no)exact:           scan ExactBoundingBox (default: false)
  --(no)debug:           debug informations     (default: false)
```



```

Examples for producing 'test.pdf':
* epstopdf test.eps
* produce postscript | epstopdf --filter >test.pdf
* produce postscript | epstopdf -f -d -o=test.pdf
Example: look for HiResBoundingBox and produce corrected PostScript:
* epstopdf -d --nogs -hires test.ps>testcorr.ps

```

### pdfcrop

You can crop a PDF file to a clipping path (e.g., its bounding box) with the program `pdfcrop`, which is also part of the T<sub>E</sub>X Live distribution.

```

> pdfcrop --help
PDFCROP 1.5, 2004/06/24 - Copyright (c) 2002, 2004 by Heiko Oberdiek.
Syntax:  pdfcrop [options] <input[.pdf]> [output file]
Function: Margins are calculated and removed for each page in the file.
Options:                                     (defaults:)
--help                                     print usage
--(no)verbose                             verbose printing          (false)
--(no)debug                               debug informations        (false)
--gscmd <name>                             call of ghostscript         (gs)
--pdftexcmd <name>                         call of pdfTeX              (pdftex)
--margins "<left> <top> <right> <bottom>"    (0 0 0 0)
                                         add extra margins, unit is bp. If only one number is
                                         given, then it is used for all margins, in the case
                                         of two numbers they are also used for right and bottom.
--(no)clip                                clipping support, if margins are set (false)
--(no)hires                                using '%%HiResBoundingBox'        (false)
                                         instead of '%%BoundingBox'
--papersize <foo>                          parameter for gs's -sPAPERSIZE=<foo>,
                                         use only with older gs versions <7.32 ()
Examples:
pdfcrop --margins 10 input.pdf output.pdf
pdfcrop --margins '5 10 5 20' --clip input.pdf output.pdf

```

## 22.4 PostScript page-manipulation tools

It is often needed to manipulate whole pages of PostScript as they come out of programs like `dvips`; one of the reasons for the Adobe File Structuring Conventions is to provide enough information to make this possible. You may wish to reorder pages, scale them, merge files together, and so on. In the next section, we look at a suite of programs that perform most of these tasks. Before that, however, let us briefly consider some of the other ways of making several logical pages print on one sheet of paper.

Many people want no more than to print “2 up”, i.e., two normal pages shrunk to fit on a single sheet of paper (in order to save paper, or to make a booklet.) Although this simple layout can be achieved with L<sup>A</sup>T<sub>E</sub>X, we shall only describe how to handle this with PostScript, since the approach is more general.

Multiple pages can be printed on a single sheet of paper by redefining in a PostScript header file the low-level PostScript `showpage` operator (it is vital that you use scaleable PostScript fonts and not bitmap .p<sub>k</sub> fonts, or PostScript's scaling will break up the characters badly.) One such header file is Ross Cartlidge's `multi.pro`; if you load this as a header file you need to put some starting PostScript code in your preamble and some closing code at the end of the pages. The `dvips start-hook` and `end-hook` procedures described in the Section 22.2.6 can be used to do this.

[illegible]

Figure 22.5: Multiple logical pages on one physical sheet, using `multi.pro`

The closing PostScript code is always `endmulti`; the starting code is the command `multi` preceded by anywhere from 3 to 7 parameters. The compulsory parameters are:

**landscape** divide the page using landscape or portrait orientation; possible values are `true` (landscape) and `false` (portrait);

rows the number of rows on each output page;

**columns** the number of columns on each output page.

The other parameters are:

**left-to-right** whether the physical page fills with logical pages left to right (`true`) or right to left (`false`); the default is `true`;

**top-to-bottom** whether the physical page fills with logical pages top to bottom (`true`) or bottom to top (`false`); the default is `true`;

**row-first**      whether to fill rows first (`true`) or columns first (`false`);

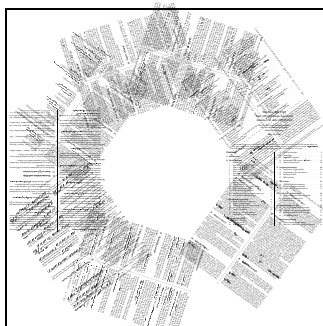
**dividers** whether to print rules between logical pages; the default is `true`.

Thus a starting code of `true 1 5 multi` places logical pages on the physical page in five columns and one row, in landscape orientation; pages are forced to fit the requested layout, so in this case they are stretched sideways to fit. The result is shown in Figure 22.5. A more useful layout would be specified by `false 4 4 multi`, four rows and four columns on a portrait page; printing 16 pages on one physical sheet page is surprisingly useful if you are just doing something like checking layout of a long book.

For really arbitrary arrangements of a set of logical pages on a physical page, you can resort to a technique we used in this book. To show an effect, we used dvips to create an EPS file of the result as

a single-page  $\text{\LaTeX}$  job and included it back in our text as a graphic. This allows ridiculous effects like printing tiny versions of the first nine pages of the  $\text{\LaTeX}$  “Frequently Asked Questions” around a circle like this (using PSTricks):

Exa.  
22-4-1



```
\usepackage{pstricks,graphicx}
\newcommand{\Page}[2]{% angle, filename
  \uput[#1]{#1}(0,0){%
    \includegraphics[width=.5in]{figures/#2}}}
\fbbox{%
  \begin{pspicture}(-2,-2)(2,2)
    \psset{labelsep=.8cm}
    \Page{360}{p1} \Page{36}{p2}
    \Page{72}{p3}  \Page{108}{p4}
    \Page{144}{p5} \Page{180}{p6}
    \Page{216}{p7} \Page{252}{p8}
    \Page{324}{p9}
  \end{pspicture}%
}
```

### 22.4.1 The psutils suite

A general set of tools to manipulate PostScript files is Angus Duggan’s `psutils`, freely available in CTAN:support/psutils. It is a set of Unix shell or Perl scripts and programs written in C that can be compiled easily on Unix and other systems. These programs are not interactive or graphical, but are controlled by command-line switches. Dimensions when required, can be specified in inches (`3in`) or centimeters (`10cm`). Many of the programs have an option to run quietly (`-q`) instead of printing page numbers as they process them, and most either read from standard input and write to standard output or have two parameters of an input file and an output file.

The programs operate on a page by page basis. Almost always, the original PostScript file must contain enough of the Adobe Document Structuring Conventions to allow the program to identify pages.

Table 22.2 on the following page gives an overview of the more useful tools available in `psutils`. The sections that follow describe the more important ones in more detail.

#### 22.4.1.1 pstops: rearranging pages in a PostScript file

```
pstops[-q] [-b] [-wwidth] [-hheight] [-ppaper] [-dlinewidth]
      pagespecs [infile] [outfile]
```

`pstops` rearranges pages in a very general way; this allows printing  $n$ -up, making booklets, reversing, scaling, etc. It is the most general tool in the set, and most users find it easier to use the higher-level `psnup` or `psbook`. The options are:

- b Prevent `bind` operators in the PostScript prolog from binding. This may be needed when complex multi-page rearrangements are being made.
- d**linewidth** Draw line of width *linewidth* around each page; if just `-d` is given, the width defaults to one point. It is important to realize that the width is relative to the size of the *original* page and is scaled. If you put four logical pages on one sheet of paper and specify `-d 2pt`, the line actually drawn is 1pt.
- h**height** The height used by the `h` dimension specifier.

Table 22.2: Tools in the psutils set

<i>Programs, Perl<sup>†</sup> and shell<sup>‡</sup> scripts</i>	
psbook	rearrange pages into signatures
psselect	select pages and page ranges
pstops	general utility for rearranging and selecting pages
psnup	combine multiple pages on a single physical sheet
psresize	alter document paper sizes
epsffit	fit an encapsulated PostScript file inside a given bounding box
getafm <sup>‡</sup>	output PostScript to retrieve the AFM file of a font from a PostScript printer
showchar <sup>‡</sup>	output PostScript to draw a character with metric info
fixfm <sup>†</sup>	fix FrameMaker documents to make psutils programs work properly
fixwfwps <sup>†</sup>	fix Microsoft Word for Windows output
fixwpps <sup>†</sup>	fix WordPerfect output
fixwwps <sup>†</sup>	fix Windows Write output
extractres <sup>†</sup>	extract resources from PostScript files
includeres <sup>†</sup>	include resources into PostScript files
psmerge <sup>†</sup>	script to merge multiple PostScript files

-p **paper** Set a named paper size; possible choices are a3, a4, a5, b5, letter, legal, tabloid, statement, executive, folio, quarto or 10x14 (the default paper size is a4).

-q Run quietly.

-w **width** The width used by the w dimension specifier.

The parameter `pagespecs` allow you to specify how pages are combined and processed, as follows:

```
pagespecs = [modulo:] specs
specs      = spec [+specs] [, specs]
spec       = [-]pageno [L] [R] [U] [@scale] [(xoff, yoff)]
```

**modulo** The number of pages in each block; must be >0 (default is 1).

**specs** The page specifications for the pages in each block. The value of `pageno` in each `spec` should be  $0 \leq$  (first page in block) and  $\leq \text{modulo}-1$  (last page in block). The optional dimensions `xoff` and `yoff` shift the page by the specified amount. These dimensions are given by default in PostScript points, but may also be specified in centimeters or inches by following them with the string `cm`, or `in`. Alternatively, one can specify the flag `w` or `h`, meaning that one is using multiples of the width or height. The optional parameters `L`, `R`, and `U` rotate the page left, right, or upside-down. The optional parameter `scale` scales the page by the specified fraction. If the optional minus sign is specified, the page number is relative to the end of the document, instead of the start.

Page *specs* separated by `+` affect pages that are to be merged into one page; page *specs* separated by a space to be on separate pages. If there is only one page specification, and `pageno=0` then `pageno` may be omitted.

The shift, rotation, and scaling are performed in sequence, regardless of the order in which they appear on the command line.

Let us consider a PostScript document `exa.ps` that contains eleven pages (see Section 22.5.3). To put two pages on one sheet of letter-sized paper, you would specify (see Table 22.1 on page 124 if you

forgot the dimensions of the standard paper sizes):

```
> pstops '2:0L@.7(8.5in,0)+1L@.7(8.5in,5.5in)' exa.ps exa2.ps
[1] [2] [3] [4] [5] [6] Wrote 6 pages, 311178 bytes
> grep Page exa2.ps
%%Pages: 6 0
%%PageOrder: Ascend
%%Page: (0,1) 1
%%Page: (2,3) 2
%%Page: (4,5) 3
%%Page: (6,7) 4
%%Page: (8,9) 5
%%Page: (10,11) 6
```

We see that the eleven pages were put onto six output sheets and by checking (with the `grep` program) for the string `%%Page` which starts PostScript structural comments we see how the pages have been combined two by two (`pstops` starts numbering pages at “0”).

Putting the same pages on A4 paper would need the following command:

```
> pstops '2:0L@.7(21cm,0)+1L@.7(21cm,14.85cm)' exa.ps exa2.ps
```

If you want to select all odd pages of our 11-page document and output them in reverse order (hence the first page is numbered `-0` in the page specification) on letter-sized paper, use:

```
> pstops '2:-0' -pletter exa.ps exareverse.ps
[1] [2] [3] [4] [5] [6] Wrote 6 pages, 253806 bytes
> grep Page exareverse.ps
%%Pages: 6 0
%%PageOrder: Ascend
%%Page: (10) 1
%%Page: (8) 2
%%Page: (6) 3
%%Page: (4) 4
%%Page: (2) 5
%%Page: (0) 6
```

The list of pages output and their order is as expected (remember you should add one to each page number shown, since `pstops` counts from zero).

### 22.4.1.2 psnup: put multiple pages on a single sheet

```
psnup[-wwidth] [-hheight] [-ppaper] [-Wwidth] [-Hheight] [-Ppaper]
      [-l] [-r] [-f] [-c] [-mmargin] [-bborder] [-dlinewidth] [-sscale]
      [-nup] [-q] [infile] [outfile]
```

`psnup` takes a PostScript file and puts several logical pages on each physical sheet of output paper. The options are:

- b **border** Leave a margin of width *border* around each (logical) page.
- c Arrange the pages down columns on the page, rather than in rows.
- d **linewidth** Draw a line of width *linewidth* around each page; if just `-d` is specified, the width defaults to 1 point. It is important to realize that the width is relative to the size of the *original* page

and is scaled. If you put four logical pages on one sheet of paper, and specify `-d 2pt` the line actually drawn is 1pt.

- `-f` Swap the page's height and width.
- `-hheight` Set the height of the output pages.
- `-Hheight` Set the height of the input pages.
- `-l` Print the output in landscape mode (rotated 90° anti-clockwise).
- `-mmargin` Leave a margin of width *margin* around the whole output page.
- `-nup` Set the number of logical pages to put on each output page. This can be any arbitrary number, and `psnup` does its best to find a suitable arrangement, but may give an error if it cannot find anything sensible.
- `-ppaper` Instead of explicitly setting the size of the output pages, this option gives a named page size; the possibilities are `a3`, `a4`, `a5`, `b5`, `letter`, `legal`, `tabloid`, `statement`, `executive`, `folio`, `quarto` or `10x14` (the default paper size is `a4` for both input and output).
- `-Ppaper` Same as `-p`, for input pages.
- `-r` Make the output print in "seascape" mode (rotated 90° clockwise).
- `-sscale` Set *scale* explicitly for page reduction. This is necessary if you have pages that are already the right size, and just want them arranged *n*-up on the output pages.
- `-width` Set the width of the output pages.
- `-Wwidth` Set the width of the input pages.

As an example of using `psutils`, let us look again at methods for putting multiple logical pages on one output page; the `psnup` program is a fast and efficient way of doing this. Figure 22.6 on the facing page shows the first page of the result of the command:

```
> psnup -9 -d exa.ps exa9.ps
[1] [2] Wrote 2 pages, 314504 bytes
```

which asks for nine logical pages on one physical page, with borders around each one.

`pstops` can do a similar job as `progsnup` but gives finer control on placing and scaling pages. For instance, to get four logical A4 pages on one physical A4 sheet of paper, you could use the command (be careful to not fragment the page specification):

```
> pstops -d2pt \
'4:0@.5(0,14.5cm)+1@.5(10.5cm,14.5cm)+2@.5(0,0.5cm)+3@.5(10.5cm,0.5cm)' \
exa.ps exa4.ps
[1] [2] [3] Wrote 3 pages, 311148 bytes
> grep %%Page exa4.ps
%%Pages: 3 0
%%PageOrder: Ascend
%%Page: (0,1,2,3) 1
%%Page: (4,5,6,7) 2
%%Page: (8,9,10,11) 3
```

We see indeed that there are now four input pages per output page. We also draw a frame around each logical page which will be 1 pt wide (since the original 2 pt will be scaled by the same factor of 0.5 as the

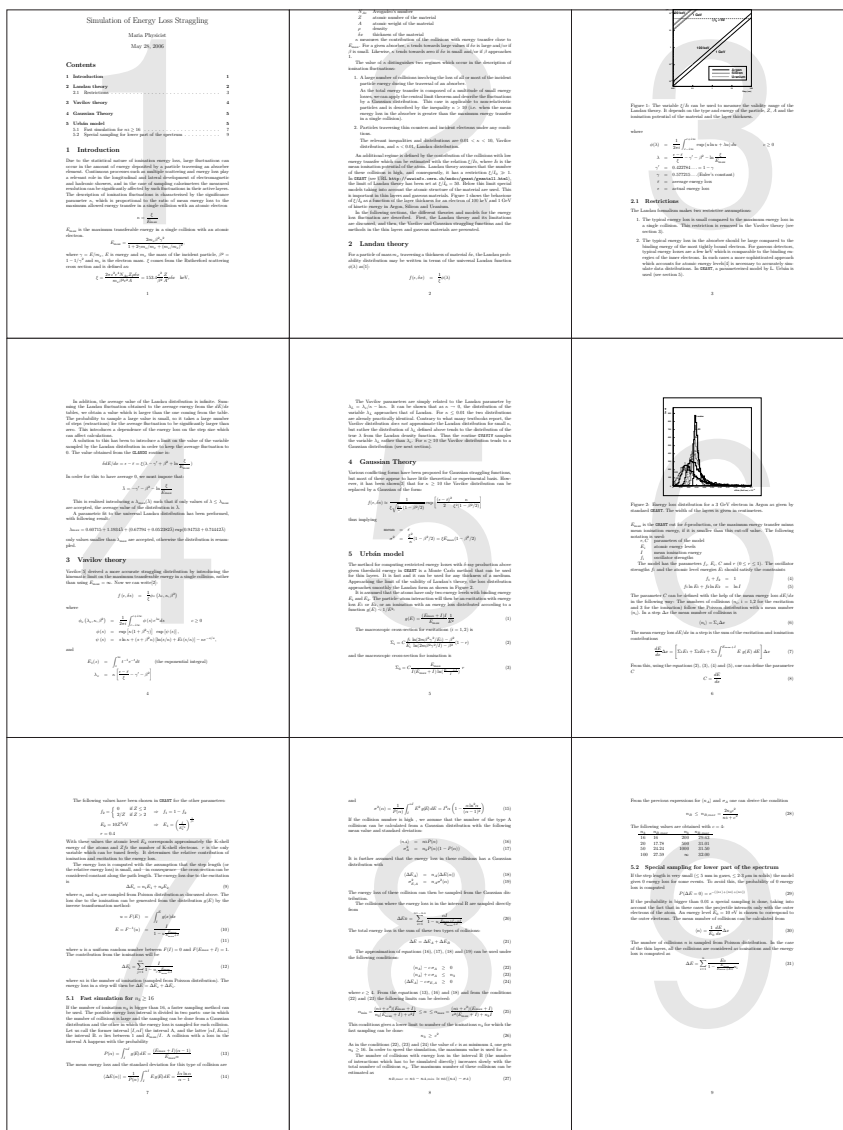


Figure 22.6: Nine logical pages on one output page

pages.

### 22.4.1.3 psbook: rearrange pages in a PostScript file into signatures

```
psbook[-q] [-ssignature] [infile] [outfile]
```

psbook takes the pages in a PostScript document and creates a new file in which the pages are rearranged as “signatures”, the sections in a printed book. In printing a book, a number of pages are printed on a large sheet of paper that is then folded and bound to make the final volume. In making a booklet by hand, it is helpful to arrange the pages so that when folded they come out in the right order. The options

to the program are:

- q Run quietly, without showing the numbers of the pages being processed.
- s **signature** Set signature size, i.e., the number of sides (a multiple of four), to be folded and bound together in the proposed book. Extra blank pages are added to make up the right number if needed.

We once more take our 11-page document and want to print it as a booklet with a signature size of 4.

```
> psbook -s4 exa.ps exabook.ps
[4] [1] [2] [3] [8] [5] [6] [7] [*] [9] [10] [11]
Wrote 12 pages, 305508 bytes
```

We notice the order in which psbook outputs the page and the blank page (marked with [\*]) it inserted to make up a multiple of four.

#### 22.4.1.4 psselect: selecting pages from a PostScript file

```
psselect[-q] [-e] [-o] [-r] [-ppages] [pages] [infile] [outfile]
```

psselect takes a subset of pages from the input file and writes a new file. The options are:

- e Select even-numbered pages only; this can be used in conjunction with the other options.
- o Selects odd-numbered pages only.
- p **pages** Specify the pages to be selected, with a comma-separated list of page ranges; these can be either a single page number or a range of two numbers separated by a colon (:). The first or last number in a range can be omitted and defaults to the first and last pages in the file. Page numbers preceded by an underscore (\_) are relative to the end of the document, counting backwards. It is important to realize that page numbers are absolute, i.e., start at one at the start of the document; the *printed* page number may be something else entirely.
- q Run quietly.
- r Output the selected pages in reverse order.

A few simple examples follow (notice in the third command how the \_ specifier counts from the back of the document).

```
> psselect -e exa.ps exaeven.ps
[2] [4] [6] [8] [10] Wrote 5 pages, 251729 bytes
> psselect -p2,5,8,11 exa.ps exa25811.ps
[2] [5] [8] [11] Wrote 4 pages, 224272 bytes
> psselect -p_1-_3,5,1 exa.ps exa11back.ps
[11] [10] [9] [5] [1] Wrote 5 pages, 223664 bytes
```

#### 22.4.1.5 psmerge: merging PostScript files

```
psmerge[-ooutfile.ps] [file1.ps file2.ps ...]
```

psmerge merges a set of PostScript files into one output file. This is quite a difficult thing to do, and can really be successful only if all the files come from the same application, since the specification of fonts



and other resources tends to vary quite widely. The options are:

**-ooutfile.ps** Specify the name of the output file.

### 22.4.1.6 psresize: scale and resize PostScript

```
psresize [-width] [-height] [-paper] [-Wwidth] [-Hheight] [-Ppaper]
        [-q] [infile] [outfile]
```

**psresize** takes an input PostScript document, rescales and centers it to fit on different-sized paper. The options are:

**-height** Set height of the output page.

**-Hheight** Set height of the input page.

**-paper** Set a named paper size for output; the possibilities are a3, a4, a5, b5, letter, legal, tabloid, statement, executive, folio, quarto or 10x14 (the default paper size is a4).

**-Ppaper** Same as **-p** but for the input page.

**-q** Run quietly.

**-width** Set width of the output page.

**-Wwidth** Set width of the input page.

For instance, if you need to print, on a printer with letter-size paper, a document that was set up for A4 paper, you can use a command like:

```
> psresize -pletter -Pa4 exa.ps exaletter.ps
[1] [2] [3] [4] [5] [6] [7] [8] [9] [10] [11]
Wrote 11 pages, 310514 bytes
```

### 22.4.1.7 Handling resource information in a PostScript file

**extractres** looks for “resources” in a PostScript document and writes them to files with the same name as the resource (with an appropriate file extension). Conversely, **includedres** physically inserts requested resources into a PostScript document. Resources are searched for in the current directory and the system default directory under the resource name.

Resources are elements of a PostScript file that are common to many different documents; the obvious examples are fonts and sets of PostScript procedures (procsets), but Level 2 PostScript also supports resources like patterns and forms. They are requested in the prologue of a PostScript document as comments of the form:

```
%%IncludeResource
```

The resources actually present in a PostScript file should be noted in a comment of the form

```
%%DocumentSuppliedResources
```

At present, the scripts described here do not change such comments, so the result is not quite perfect.

A possible application for these utilities is to rewrite a document containing several copies of resources so that it has just one copy of each in a clean way. Using Unix pipes, you can perform this job with a command like:

```
> extractres file1.ps | includeres > file2.ps
```

You could also use `extractres` to downsize files for sharing with colleagues by email, if you are sure they have the same fonts and procsets to re-include when they receive your file.

The programs are:

```
extractres[-m]< infile.ps > outfile.ps
includeres    < infile.ps > outfile.ps
```

-m Combine resources with the same name into a single file.

We can extract the resources `dvips` has put in the file `exa.ps` and then list them as follows:

```
> extractres -m < exa.ps >exares.ps
> grep -i Resource exares.ps
%%IncludeResource: procset tex.pro 0 0
%%IncludeResource: procset texps.pro 0 0
%%IncludeResource: procset special.pro 0 0
%%IncludeResource: font CMSY10
%%IncludeResource: font CMR8
%%IncludeResource: font CMMI12
%%IncludeResource: font CMMI5
%%IncludeResource: font CMR5
%%IncludeResource: font CMTI10
%%IncludeResource: font CMBX12
%%IncludeResource: font CMSY7
%%IncludeResource: font CMEX10
%%IncludeResource: font CMTT10
%%IncludeResource: font CMMI7
%%IncludeResource: font CMR7
%%IncludeResource: font CMMI10
%%IncludeResource: font CMR10
%%IncludeResource: font CMBX10
%%IncludeResource: font CMR12
%%IncludeResource: font CMR17
```

We note the three header files of `dvips`, where it specifies all the utility definitions to optimize the PostScript it generates, plus all the font instances (all Computer Modern fonts used in the eleven pages of the document).

#### 22.4.1.8 epsffit: fitting EPS files into a constrained size

```
epsffit[-c] [-r] [-a] [-m] [-s] llx lly urx ury [infile] [outfile]
```

`epsffit` puts suitable PostScript code in an EPS file to make it fit into a new bounding box. The bounding box is specified in the form `llx lly urx ury`, where `llx lly` are the  $x$  and  $y$  coordinates (in PostScript points) of the lower left corner and `urx ury` of the upper right corner. The options are:

-a Adjust the aspect ratio to fit the bounding box (by default the aspect ratio is preserved).

- c Center image in the bounding box.
- m Rotate image to maximize its size if that would fit the bounding box better.
- r Rotate image by 90 degrees counterclockwise.
- s Add a `showpage` command at the end of the output file to force the image to print.

### 22.4.2 Adding labels to included pictures with `psfrag`

The `psfrag` package by Michael Grant, Craig Barratt, and David Carlisle solves a common problem in graphics file inclusion. It provides a mechanism whereby  $\TeX$  can be made to typeset labels, equations, and other complex material that override the textual elements in a PostScript graphic file. It works by getting the PostScript interpreter to identify strings in your output file, and then providing replacements.

The `psfrag` package defines the command `\psfrag` and it allows you to “tag” a word in the PostScript file at the position where you want  $\TeX$  material to be typeset. The replacement for the tag in the  $\TeX$  document is defined by the arguments of the command.

```
\psfrag{tag}[posn][psposn][scale][rot]{ $\TeX$  material}
```

The `tag` is replaced by the  $\TeX$  material. The first two optional arguments are used to define how the bounding boxes of the  $\TeX$  text and the PostScript text line up, using the same syntax as the standard `\makebox` command. The third and fourth optional arguments `scale` and `rot` change the scaling and rotation of the typeset text.

As an example, consider the following PostScript file:

```
%!
%%BoundingBox:100 100 172 136
1 setlinewidth
100 100 moveto
/Times-Roman findfont
12 scalefont setfont
(nabla) show
100 120 moveto
(ALPHA) show
showpage
```

This produces:

ALPHA

nabla

when processed normally. If you put the command:

```
\psfrag{ALPHA}{ $\alpha$  \mathcal{ALPHA}}
\psfrag{nabla}{ $\nabla$ }
```

in the  $\TeX$  file that includes the graphic and run it again, you get:

$\alpha\mathcal{ALPHA}$

▽

All `\psfrag` calls that precede an `\includegraphics` command (or equivalent) are used for all subsequently included PostScript files. This permits the definition of global as well as local `\psfrag` substitutions.

This method requires no editing of the PostScript file, as it is all done by the PostScript interpreter; it does, however, assume certain things about the way text is presented in the PostScript output from a program. Some packages may treat every letter of a word as a separate string to output, while even in the simple output from programs like `dvips`, it is not obvious which words come through as complete strings. For instance, while most of the words in the following piece of PostScript are amenable to processing by `psfrag`, several are broken in unexpected places:

```
118 w(Y)l(ellowGreen)p SpringGreen 1734 1156 V Black
104 w(SpringGreen)p OliveGreen 46 1255 V Black 85 1305
a(OliveGreen)p RawSienna 384 1255 V Black 127 w(RawSienna)p
Sepia 721 1255 V Black 179 w(Sepia)p Brown 1059 1255
V Black 226 w(Brown)p Tan 1396 1255 V Black 245 w(T)g(an)p
Gray 1734 1255 V Black 260 w(Gray)p Black Black eop
```

It is also possible to get unwanted effects—if you mark the string “A” as a tag to be translated as  $\alpha$ , then all distinct strings which the generating program puts out as “A” are translated. On the other hand, if you tag “A” as  $\alpha$  and “B” as  $\beta$ , then “AB” is *not* translated.

If the included graphics file is to be resized on inclusion, it is important to understand the distinction between:

```
\resizebox{2in}{!}{\includegraphics{file.ps}}
```

and (using the `graphicx` package):

```
\includegraphics[width=2in]{file.ps}
```

Using the current graphics packages drivers, in the former case `psfrag` elements are scaled with the drawing, and in the latter case they are not. Since both sorts of behavior may be needed at different times, this is to be regarded as a feature, not a problem.

`psfrag` is principally useful in conjunction with fairly simple drawing programs such as `xfig` or `idraw` and plotting packages like `MATLAB` or `gnuplot`. In the latter programs, strings can typically be specified for axis labels on graphs and can be set by specifying unique tag identifiers and defining the  $\text{\TeX}$  replacements. When a large number of plots are being prepared in some analysis program, then `psfrag` lets you automate typesetting the text. However, although `psfrag` is a clever piece of work that does its job well, it is not ideal for processing large-scale, arbitrarily complicated PostScript. The PostScript file should ideally be designed *with psfrag in mind*. Note, also, that systematic use of `psfrag` requires a good understanding of both the PostScript language and the application generating the figures.

## 22.5 Producing PDF from various sources

The only graphical object which  $\text{\TeX}$  can handle internally is the `picture` environment, which is very easy to use, but rather restrictive. All other graphical material must be encapsulated in `\special` commands and later extracted by the DVI processor, e.g., `dvips`, and transformed into PostScript code.

Packages like `pstricks` (and its extensions `pst-xxxx`) and `psfrag` can produce such `\special` commands. Unfortunately, `pdflatex` cannot work directly with PostScript files. Depending on the presence of EPS graphics files to be included by  $\TeX$  one or more strategies can be used to obtain PDF output:

- traditional: `latex` creates a DVI file which is read by `dvips` which creates a PostScript file which is finally translated into PDF by `ps2pdf`;
- using `dvipdfm` to skip the PostScript-generating step;
- using `pdflatex` to skip the DVI step and generate PDF directly;
- using the package `pst-pdf`;
- using one of the commercial packages that have been mentioned in Section 22.2 (e.g., `VT $\TeX$` ) and which generate PDF directly.

In this section we first describe the `dvipdfm` and `dvipdfmx` programs, which generate PDF from a DVI file. Then we turn our attention to the `pst-pdf` package, which automates the translation of EPS images and `PSTricks` PostScript code into PDF. The final part of the section looks at an example of a  $\TeX$  file that is translated into PDF using each of the first four alternatives mentioned above.

### 22.5.1 The programs `dvipdfm` and `dvipdfmx`

Mark A. Wicks's program `dvipdfm` (<http://gaspra.kettering.edu/dvipdfm/>) supports the following features:

- bookmarks, named destinations, and annotations (many of Acrobat Distiller's `pdfmark` features);
- `dvips` specials;
- inclusion of MetaPost output and of arbitrary PostScript files with help from an external program;
- thumbnails (generated by Ghostscript);
- arbitrary, nested linear transformations (including scaling and rotation) of typeset material;
- inclusion of PDF images, including cropping by supplying a bounding box.
- inclusion of JPEG and PNG images;
- a color stack for keeping track of the current color;
- partial font embedding and Flate compression to reduce file size;
- balancing of page and destination trees to speed up reader access for very large documents.

A detailed description of how these functions are supported can be found in the program documentation (CTAN:dviware/dvipdfm/dvipdfm-0.13.2c.pdf). In order to take advantage of these functions when running `dvipdfm` on a DVI file you must specify the `dvipdfm` option with  $\TeX$  (and `hyperref`).

A few years ago the author of `dvipdfm` ceased active development. In the meantime, around 2001 Shunsaku Hirata had developed a variant, `dvipdfm-jpn`, for Japanese, and Jin-Hwan Cho a variant, `dvipdfm-kor`, for Korean. In 2002 they decided to combine their efforts to further enhance the Unicode capabilities of `dvipdfm`, adding support for multi-byte character encodings and large character sets for East Asian languages by CID-keyed fonts. They released jointly the `dvipdfmx` program (<http://project.ktug.or.kr/dvipdfmx/>), which also has support for many features of Hàn Thê Thành's `pdf $\TeX$`  program.

`dvipdfmx` is a must if one wants to deal with large character sets, since all traditional methods, especially `pdf $\TeX$` , cannot handle those natively. For instance `dvipdfmx` lets you extract and search 16 bit characters. Full support for PostScript Type 1, TrueType, but also OpenType is provided, and if the font resides on the system one can instruct `dvipdfmx` not to embed it. PDF encryption and multiple pagesizes in a single document are possible.

The command line options of `dvipdfm` are the following:

- c**     *Disable color specials.*  
This option forces all color commands to be ignored. Useful for printing a color document on a black and white printer.
- e**     *Disable partial font embedding.*  
Useful for forms which need complete fonts, or for PFB files that `dvipdfm` cannot parse.
- f**     *Set font map ffile name* (default `tlfonts.map`).
- l**     *Select Landscape.* Only meaningful for paper sizes specified on the command line.
- m number**     *Specify additional magnification for document.*
- o filename**     *Output PDF ffile name* (default `dvifile.pdf`).
- p papersize**     *Output papersize* (default “letter”).  
Possible other values are “legal”, “ledger”, “tabloid”, “a4”, and “a3”; *papersize* can also be specified as *w*<unit>, *h*<unit>, e.g., “20cm, 30cm”.
- s page\_ranges**     *Selects subset of pages from the DVI ffile.*  
Similar to `dvips`’s `-pp` option, but with the colon range indicator replaced by a hyphen, e.g., `dvipdfm -s 10,21,73-92` prints pages 10, 21 and 73 through 92. If the first page in a range is empty, PDF generation starts at the beginning of the document (`dvipdfm -s -20`), while if the last page in a range is empty the end of the document is taken (`dvipdfm -s 97-`).
- t**     *Embed thumbnail images.* Thumbnails must be generated by a separate program.
- d**     *Delete thumbnail images after embedding.*
- x number**     *Horizontal offset for document* (default 1 in).
- y number**     *Vertical offset for document* (default 1 in).
- z number**     *zlib compression level.*  
*number* in range 0 (no compression) to 9 (maximal compression, the default)
- v**     *Verbose.* Display complete file
- vv**     *Superverbose.* Display maximal log messages.

To the above `dvipdfm` adds the following options:

- d number**     *PDF decimal digits.* *number* in range 0–5 (default 2).
- r number**     *Resolution for raster fonts.* In DPI (default 600).
- C number**     *Option flags* (default 0).  
  - 0x0001**     reserved;
  - 0x0002**     use semi-transparent filling for `tpic` shading command, instead of opaque gray color (requires PDF 1.4);
  - 0x0004**     treat all CIDFonts as fixed-pitch fonts;
  - 0x0008**     do not replace duplicate fontmap entries.

Positive values are always *ORed* with previously given flags, while negative values replace old values.
- O number**     *Maximum depth of open bookmark items* (default 0).

- P *number*** *Permission flags for PDF encryption* (default 0x003C).
- S** *Enable PDF encryption.*
- T** *Embed thumbnail images.* Like `-t`, but image files are removed when finished.
- V *number*** *PDF minor version* (default 3).

## 22.5.2 From PostScript to PDF with pst-pdf

The `pst-pdf` package uses the `ETEX` package `preview`, which is part of the `preview` [8, 16] bundle. `preview` extracts all “marked” parts in a `ETEX` document into a DVI file, in which each such part is saved on a separate page. This makes it easy to convert this DVI file into PDF format and then include these parts in a subsequent `pdflatex` run.

### 22.5.2.1 Package options

<code>active</code>	enables the extraction modus of the <code>preview</code> package; the DVI output collects only the images (default);
<code>inactive</code>	only the packages <code>pstricks</code> and <code>graphicx</code> are loaded, all macros are disabled;
<code>pstricks</code>	package <code>pstricks</code> is loaded (default);
<code>nopstricks</code>	package <code>pstricks</code> is not loaded; however, if the macro detects any <code>PSTricks</code> macro, then <code>pstricks</code> will be loaded automatically nevertheless;
<code>draft</code>	same meaning as for package <code>graphicx</code> , but only valid for the last <code>pdflatex</code> run;
<code>final</code>	in the last <code>pdflatex</code> mode the container file is used (default);
<code>tightpage</code>	whitespace around images is cut (default);
<code>notightpage</code>	whitespace around images is not cut;
<code>displaymath</code>	treats <code>displaymath</code> , <code>eqnarray</code> , <code>equation</code> , and <code>\$\$</code> or <code>\(...\)</code> as images;
<code>other</code>	all other options are passed to the package <code>pstricks</code> .

When you specify the `inactive` option all the `pst-pdf` macros will be disabled, apart from the trimming function, so that `latex` can be run in the usual way and PostScript output can be generated (with `dvips`), if desired.

### 22.5.2.2 Usage

`pst-pdf` was first designed for `PSTricks`. This is why it supports by default the `pspicture` and `psmatrix` environments, as well as all macros which are defined as `\pst@object`. `pst-pdf` works via the package `preview` completely in the background and one only has to load the package in the preamble of a document. The process of generating a PDF file from a `ETEX` source consists of two stages: the creation of the graphics container and the subsequent `pdflatex` run to create the PDF. These stages are described next.

#### *Creation of the graphics container*

##### **latex *file.tex***

Initial run of `latex`, where `preview` extracts all known objects, and saves them into `file.dvi`, where each object is on its own page. The DVI file thus created is of a special internal format and is unsuited for user purposes, such as viewing with a DVI viewer.

**dvips -Ppdf -o file-pics.ps file.dvi**

dvips run to convert the DVI file to PostScript, where the `-Ppdf` option tells dvips to load the config file for PDF-related output. dvips creates the new file `file-pics.ps`.

**ps2pdf file-pics.ps file-pics.pdf**

ps2pdf run to convert the PostScript file to PDF, with each image on a separate page.

### *Creation of the final PDF output document*

**pdflatex file.tex** First run of pdflatex run where pst-pdf is not active.

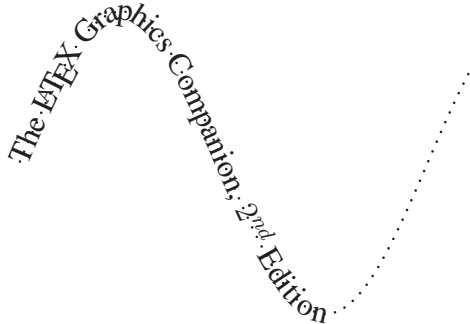
**bibtex file** bibtex run.

... Any other additional runs (e.g., index, glossary).

**pdflatex file.tex** Ultimate pdflatex run, where all generated PDF images are included.

A simple example with PSTricks follows.

This is a **PDF**-document!



```
\usepackage{pst-plot,pst-text}
\usepackage{pst-pdf}
This is a \textbf{PDF}--document!

\begin{pspicture}(-0.25,-2.25)(6.25,2.5)
\pstextpath[linestyle=none]%
{ \psplot[linewidth=1pt,%
  linestyle=dotted,%
  plotpoints=300,%
  xunit=0.015,%
  yunit=2]{0}{400}{x sin}}
{ \large The \LaTeX\ Graphics
  Companion, $2^{nd}$ Edition}
\end{pspicture}%
```

Exa.  
22-5-1

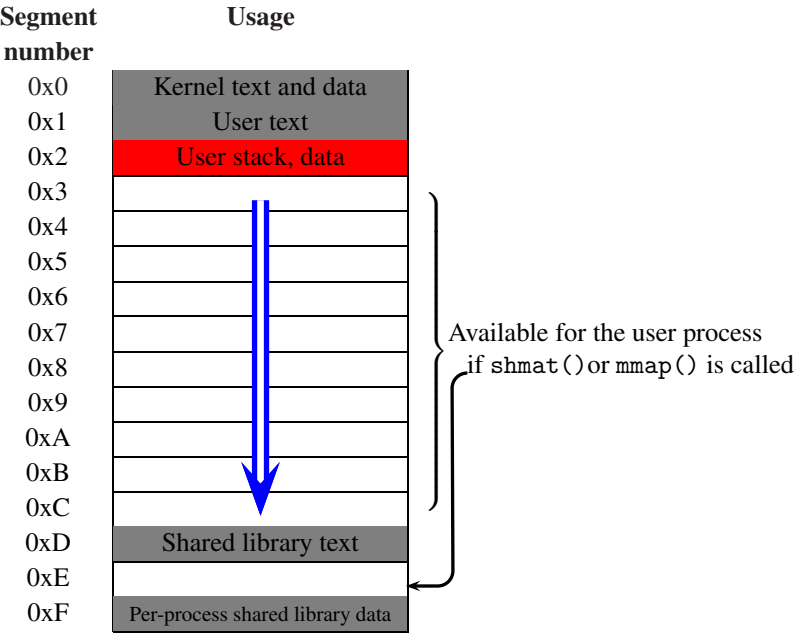
pst-pdf provides a macro `\PreviewEnvironment`, which lets you define additional environments, which are scanned by the preview package and are also written as an image into the DVI file. In the following example, PSTricks is used to connect some nodes in a tabular. With the command `\PreviewEnvironment{tabular}` this environment is also written into the DVI file. There are no restrictions in declaring environments for preview.

```
\usepackage{bigdelim,multirow,array}
\usepackage[table]{pstricks}
\usepackage{pst-node,pst-pdf}
\PreviewEnvironment{tabular}
\definecolor{Gray}{gray}{0.1}
\renewcommand\arraystretch{1.1}
\begin{tabular}{c|c|l}
\multicolumn{1}{c}{\textbf{Segment}} & \multicolumn{1}{c}{\textbf{Usage}} & \\
\multicolumn{1}{c}{\textbf{number}} & \multicolumn{1}{c}{\textbf{}} & \\
0x0 & \cellcolor{gray}Kernel text and data & \\
0x1 & \cellcolor{gray}User text & \\
0x2 & \cellcolor{red}User stack, data & \\
0x3 & \pnode{A} & \\
& \rdelim}{10}{5.5cm}[\parbox{7.5cm}{Available for the user process\\
& \hspace*{0.25cm}\pnode{A2}if \texttt{shmat()}or \\
\texttt{mmap()} is called}]] & \\
0x4 & & 0x5 & & \\
\end{tabular}
```



```
0x6 & & \\\cline{2-2} 0x7 & & \\\cline{2-2}
0x8 & & \\\cline{2-2} 0x9 & & \\\cline{2-2}
0xA & & \\\cline{2-2} 0xB & & \\\cline{2-2}
0xC & \pnode{B} & \\\cline{2-2}
0xD & \cellcolor{gray}Shared library text & \\\cline{2-2}
0xE & & \pnode{B2} & \\\cline{2-2}
\ncline[arrows=->,linewidth=2pt,linecolor=blue,doubleline=true]{A}{B}%
\ncdiag[arrows=->,linewidth=1.25pt,linearc=0.2,%
angleA=180,angleB=0,armA=0.2cm,nodesepB=-0.25cm,armB=0.625cm]{A2}{B2}%
0xF & \cellcolor{gray}\footnotesize Per-process shared library data & \\\cline{2-2}
\end{tabular}
```

Exa.  
22-5-2



The package `pst-pdf` supports also `psfrag` (see Section 22.4.2) and EPS images with the help of a `postscript` environment, whose contents is scanned by `preview` and written into the DVI file and then converted to PDF. This is sometimes easier than using `ps2pdf`, because the conversion of the EPS image occurs in the background.

It is important to realize that `pst-pdf` numbers all images consecutively. If anything changes in the order of the images, when an image is added, deleted, or just edited, the first three runs for building the graphics image container have to be repeated. On the other hand, if only the text was edited, then rerunning `pdflatex` once is sufficient, as long as the PDF image container exists, since all images are taken from there.

### 22.5.3 Generating PDF from $\text{\LaTeX}$

As explained at the beginning of this section, there are various ways of generating a PDF file from a  $\text{\LaTeX}$  source. The route that you can follow depends mostly on the graphics material that you want to include. If most of it is in EPS format, the easiest way is to use `latex`, followed by `dvips` and finally `ps2pdf`. If you have all files already in PDF format, with some JPEG and PNG images, the more direct route is to run `pdflatex`. You can also combine both approaches by running `latex` and the `dvipdfmx` program. And, of course, you can use the technique introduced in Section 22.5.2 based on the `pst-pdf` package.

As an example of these four possibilities we use a medium-sized file `exa.tex`, where we also are interested in taking advantage of PDF's hypertext capabilities by loading the `hyperref` package in the  $\text{\LaTeX}$  source. As the way the  $\text{\LaTeX}$  structural information is translated into PDF hypertext commands differs for each program (`dvips`, `dvipdfm`, and `pdflatex`), we have to indicate which program will generate the final PostScript or PDF output (see the three first lines of the  $\text{\LaTeX}$  source of the file `exa.tex`).<sup>1</sup>

```
\documentclass[a4paper,dvipdfm]{article}
%\documentclass[a4paper,dvips]{article}
%\documentclass[a4paper,pdftex]{article}
\usepackage{graphicx}
\usepackage{url}
\usepackage{makeidx}
\usepackage[backref]{hyperref}
\makeindex
\title{Simulation of Energy Loss Straggling}
\author{Maria Physicist}
\begin{document}
\maketitle
\tableofcontents

\section{Introduction}
\index{Straggling}
```

### Running the example with `latex`, and `dvipdfmx`

For the first run we use `dvipdfmx` to generate the PDF. Therefore we must ensure that we have the images also as `.pdf` files and that each image is accompanied by a small text file that specifies its bounding box (`dvipdfmx` assumes that for each image `fig.pdf` there exists a file `fig.bb`). For transforming EPS files into PDF we can use the program `epstopdf` (see page 134). Information about the resulting PDF file can be obtained with the `pdftinfo` utility (part of the `xpdf` distribution) as follows.

```
> ls *.eps
phys332-1.eps  phys332-2.eps
> more phys332-1.bb
%%BoundingBox: 0 0 567 567
> epstopdf phys332-1.eps
> pdftinfo phys332-1.pdf
Producer:      GNU Ghostscript 7.05
Tagged:        no
Pages:         1
Encrypted:     no
Page size:     567 x 567 pts
File size:     11549 bytes
Optimized:     no
PDF version:   1.3
```

We observe that the bounding box of the PDF corresponds to the one of the EPS source. If this were not the case, the PDF image can be cropped to the correct size with the `pdfcrop` utility (see page 135).

<sup>1</sup>This example is basically identical—a few `hyperref` and PDF-related lines have been added—to the  $\text{\LaTeX}$  code described in Appendix A of the *LaTeX Web Companion*[6] and is available as `info/examples/lwc/apa/latexexa.tex` on CTAN.

Next we run the  $\text{\LaTeX}$  source `exa.tex` the correct number of times through `latex`, before generating the PDF file with `dvipdfmx`, as follows.

```
> latex exa
> latex exa
> makeindex exa
> latex exa
> dvipdfmx -o exadvipdfmx.pdf exa
exa.dvi -> exadvipdfmx.pdf
[1] [2] [3] [4] [5] [6] [7] [8] [9] [10] [11]
130030 bytes written
```

The resulting file is written to `exadvipdfmx.pdf`, and can be viewed with `ghostview`, `Acrobat Reader`, etc. Since we activated the `hypertext` package in the  $\text{\LaTeX}$  source, the viewer can navigate conveniently through the document.

### Running the example with `latex`, `dvips` and `ps2pdf`

If we activate the `dvips` option on the `\documentclass` command in the  $\text{\LaTeX}$  source and run `latex` the correct number of times, we can use `dvips` and `ps2pdf` (or its explicit variant `ps2pdf13`) to obtain the PDF output file `exadvips.pdf`. This file has the same (hypertext) characteristics as `exadvipdfmx.pdf` of the previous example.

```
> rm *.aux      # get rid of program-specific entries in aux file
> latex exa
> latex exa
> makeindex exa
> latex exa
> dvips -j0 exa -oexadvips.ps
> ps2pdf13 -sPAPERSIZE=a4 exadvips.ps
```

### Running the example with `pdflatex`

Since we already all the images in PDF format we can directly run `pdflatex`, and obtain the file `exapdflatex.pdf`, which is functionally equivalent to the PDF output files generated in the two previous cases.

```
> rm *.aux      # get rid of program-specific entries in aux file
> pdflatex exa
> pdflatex exa
> makeindex exa
> pdflatex exa
> mv exa.pdf exapdflatex.pdf
```

### Running the example with `pdflatex`, using the `pst-pdf` package

As the fourth alternative for generating PDF, we load the `pst-pdf` package in the preamble of our example  $\text{\LaTeX}$  file `exa.tex`, so that we now have (see page 152):

```
\documentclass[a4paper,dvips]{article}
%\documentclass[a4paper,pdftex]{article}
\usepackage{graphicx}
```

```
\usepackage{url}
\usepackage{makeidx}
\usepackage{pst-pdf}%<<<<< line added
\usepackage[backref]{hyperref}
```

First we produce the PDF version of the EPS images by running with the `dvips` option for the hyperlinks (see Section 22.5.2 for the details of the procedure).

```
> latex exa
...
Output written on exa.dvi (2 pages, 3344 bytes).
Transcript written on exa.log.
> dvips -Ppdf -o exa-pics.ps exa.dvi
This is dvips(k) 5.95b Copyright 2005 Radical Eye Software
' TeX output 2006.05.30:1632' -> exa-pics.ps
<tex.pro><alt-rule.pro><pstricks.pro><pst-dots.pro><special.pro>.
[1<phys332-1.eps>] [2<phys332-2.eps>]
> ps2pdf exa-pics.ps exa-pics.pdf
```

The file `exa-pics` now contains the two PDF instances of the EPS pictures referenced in the original document. As we are now going to run with `pdflatex`, we activate the line with the `pdftex` option on the `\documentclass` command, and process the file `exa.tex` the relevant number of times, with, if needed, runs of `makeindex` and `bibtex` interspersed to generate index and bibliographic references.

```
> pdflatex exa
> makeindex exa
> pdflatex exa
> pdflatex exa
```

In summary (see Figure 22.7 on the next page), when deciding which method to use to generate PDF output starting from a  $\text{\LaTeX}$  source file, one can state that the `latex`  $\rightarrow$  `dvips`  $\rightarrow$  `ps2pdf` route is appropriate for cases where most of the external graphics files are in EPS format. When a lot of PSTricks images are present in the source the use of the `pst-pdf` package and `pdflatex` is to be seriously considered. The more direct `pdflatex` route seems more attractive if the graphics files are available as `.pdf`, `.jpeg` or `.png` files. Finally, the choice of `dvipdfmx` only seems necessary if large, multi-byte font sets (e.g., for handling Far-East Asian languages) are required. The PDF files generated by the four methods discussed in the current section are completely functionally equivalent.

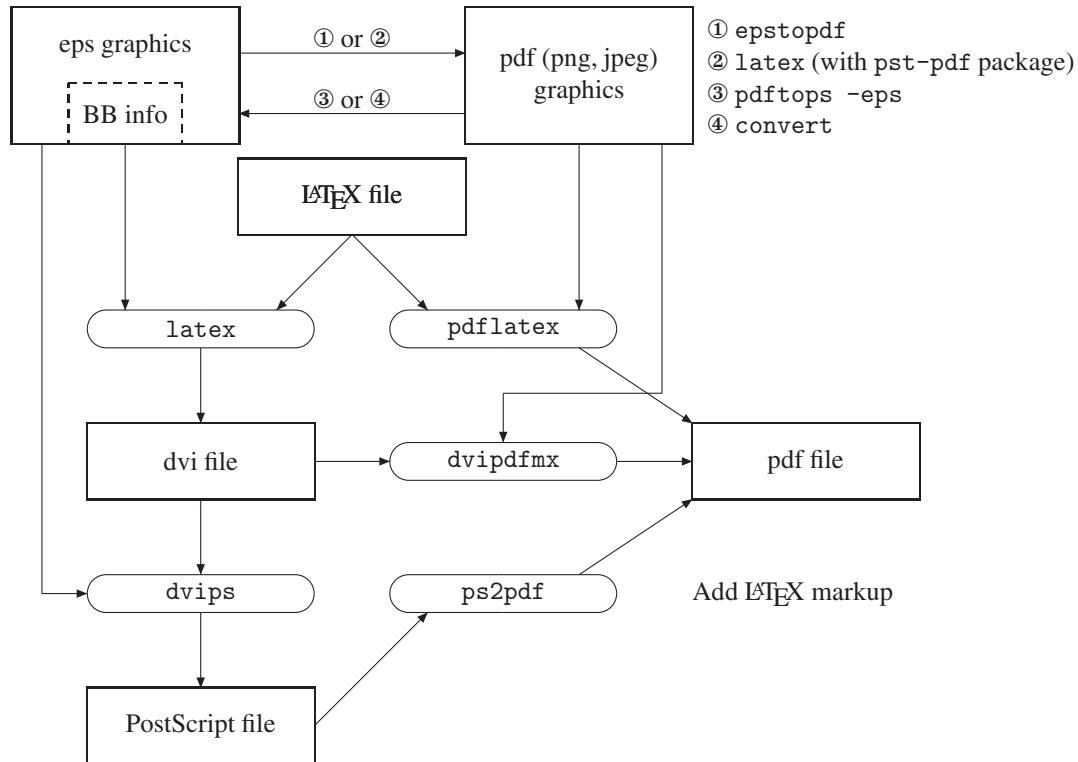
As an example of what the generated PDF output looks like, Figure 22.8 on page 156 shows the first page of the file `exapdflatex.pdf` as displayed by a PDF viewer (the three other PDF files would show a fully identical screen).

## 22.6 PDF manipulation tools

### 22.6.1 pdftk

`Pdftk` (<http://www.accesspdf.com/pdftk/>) is a freely available command-line tool which runs on most computer platforms and which can perform the following tasks on PDF documents:

- merge PDF documents;
- split PDF pages into a new document;
- rotate PDF pages or documents;

Exa.  
22-5-3Figure 22.7: Various ways to generate PDF from  $\LaTeX$ .

- decrypt input as necessary (password required);
- encrypt output as desired;
- fill PDF forms with PDF data and/or flatten forms;
- apply a background watermark or a foreground stamp;
- report on PDF metrics such as metadata, bookmarks, and page labels;
- update PDF metadata;
- attach files to PDF pages or the PDF document;
- unpack PDF attachments;
- burst a PDF document into single pages;
- uncompress and re-compress page streams;
- try and repair corrupted PDF.

The list of available options is displayed as follows.

```
> pdftk -h
pdftk <input PDF files | - | PROMPT>
[input_pw <input PDF owner passwords | PROMPT>]
[<operation> <operation arguments>]
[output <output filename | - | PROMPT>]
[encrypt_40bit | encrypt_128bit]
[allow <permissions>]
[owner_pw <owner password | PROMPT>]
```

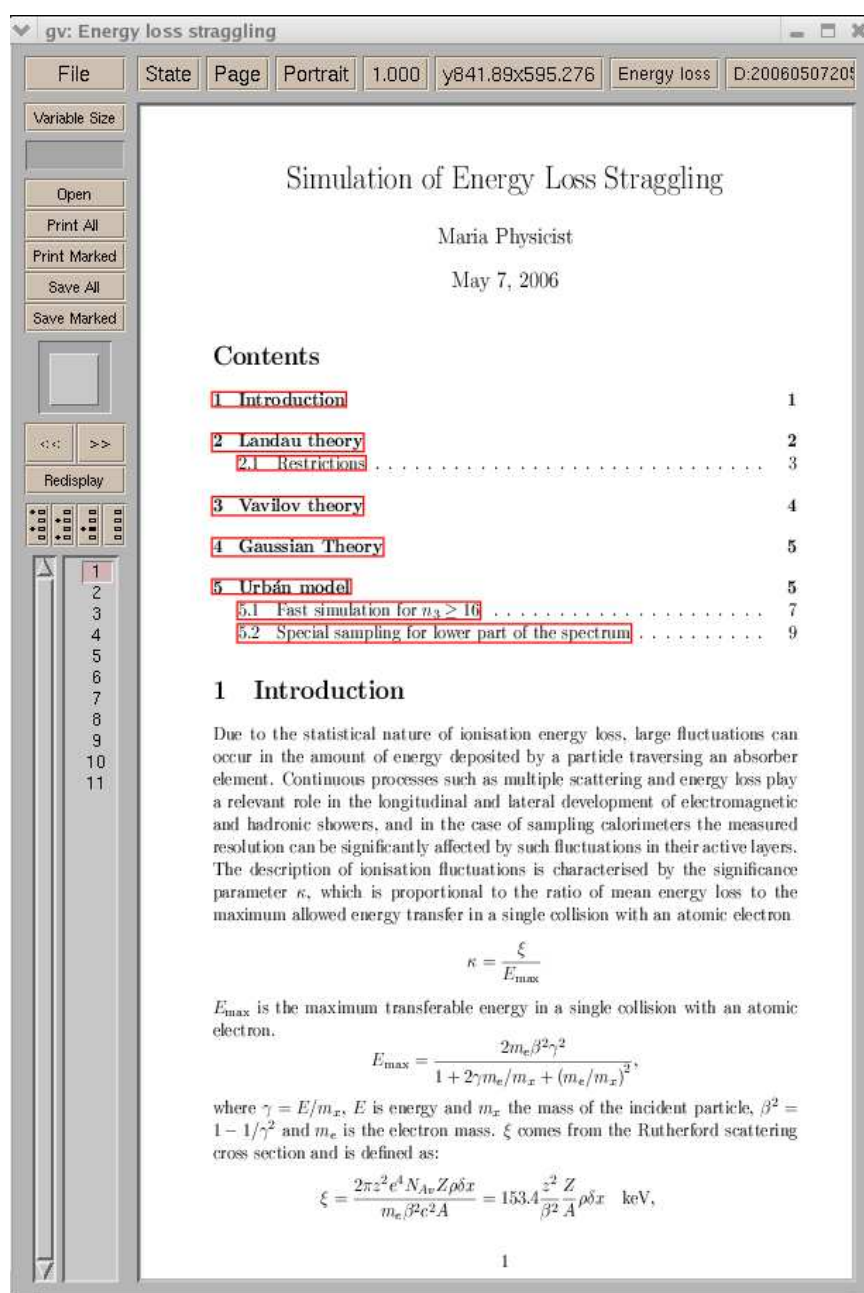


Figure 22.8: Hypertext document generated with pdf<sub>l</sub>at<sub>ex</sub>. The first page of the file `exapdflatex.pdf` is displayed with ghostview. The hyperlinks (surrounded by boxes) allow one to jump from the various entries in the table of contents to the start of the various sections in the document body. We shall refer to this first page of the file `exa.pdf` on various occasions in the description of the PDF manipulation tools (Section 22.6).

```

[user_pw <user password | PROMPT>]
[flatten] [compress | uncompress]
[keep_first_id | keep_final_id] [drop_xfa]
[verbose] [dont_ask | do_ask]
Where:
<operation> may be empty, or:
[cat | attach_files | unpack_files | burst |
 fill_form | background | stamp | generate_fdf
 dump_data | dump_data_fields | update_info]

```

A short description of the options follows.

#### **<input PDF files | - | PROMPT>**

List of the input PDF files. If no handles are defined the files should be listed in the order that they have to be combined. A single PDF file can be read from standard input (specify - as first argument to pdftk if you want this).

```

# The files f1.pdf, f2.pdf and f3.pdf are merged into f123.pdf
pdftk f1.pdf f2.pdf f3.pdf cat output f123.pdf

```

Input files can be associated with handles, consisting of a single, upper-case letter, e.g.,

```

# Using handles for merging files
pdftk F=f1.pdf G=f2.pdf cat F G output f12.pdf

```

Handles are especially useful when specifying PDF passwords or page ranges.

#### **[input\_pw <input PDF owner passwords | PROMPT>]**

Input PDF owner passwords, if needed, are associated with files by their handles:

```

<input PDF handle>=<input PDF file owner password>

```

```

# Using handles to specify passwords
pdftk A=a.pdf B=b.pdf input_pw A=apw B=bpw cat A B output ab.pdf
# Join two files, the 1st file has a password, the output has none
pdftk A=secure.pdf b.pdf input_pw A=secret cat output both.pdf

```

Without handles passwords are associated with input files by order.

#### **[<operation> <operation arguments>]**

Without this optional argument pdftk runs in “filter” mode, which takes only one PDF file as input and creates a new PDF file after applying all of the output options, such as encryption and compression.

The available operations are described next.

#### **cat [<page ranges>]**

Concatenates pages from the input PDF files to create a new PDF file, with its page order specified by the order of the given page ranges. Page ranges have the following format:

```

<input PDF handle>[<begin page>[-<end page[even|odd]]][<page rotation>]

```

The handle identifies one of the input PDF files, and the beginning and ending page numbers are one-based references to pages in the PDF file. The page rotation can be N, S, E, W, L, R, or D.

The *even* (or *odd*) qualifier causes `pdftk` to use only the even-numbered (odd-numbered) PDF pages, so `1-6even` yields pages 2, 4, and 6 in that order. `6-1even` yields pages 6, 4, and 2 in that order.

Specifying a page rotation causes `pdftk` to rotate pages and documents using the following conventions.

*absolute orientations*

N: 0°, E: 90°, S: 180°, W: 270°;

*relative orientations*

L: -90°, R: +90°, D: +180°.

If the handle is omitted from the page range, then the pages are taken from the first input PDF. If no arguments are passed to `cat`, then `pdftk` combines all input PDF files in the order they were specified to create the output.

- `<begin page>` may be larger than `<end page>`, e.g., `A1-21`
- the final page in a document can be specified using the `end` keyword, e.g., `Bend`
- a single page in a document can be specified by omitting the ending page number, e.g., `A4`
- an entire document can be included by just specifying its handle, e.g., `A` is the same as `A1-end`

*specifying page ranges without handles*

**1-endE** entire document rotated by 90°  
**5 11 20** extract the three pages given  
**5-25oddW** extract the take odd pages in range and rotated by 90°  
**6-1** extract first six pages in reverse order

*specifying page ranges with handles* Suppose `A=in1.pdf` `B=in2.pdf`

**A1-21 Beven** extract first 21 pages from A then even pages from B  
**A1-15 Bend-1odd Aend** extract first 15 pages from A then odd pages from B in reverse order, then add last page of A  
**AW BE** extract all pages of A rotated by 270° followed by all pages of B rotated 90°  
**A2-30evenL** extract even pages from range and turning each page 90° backwards  
**AevenW Aodde** extract even pages of A rotated by 270° followed by its odd pages rotated 90°  
**AW BW BD** extract all pages of A and B rotated by 270° followed by all pages of B turned upside-down.

**attach\_files** `<filenames | PROMPT>` [`to_page` `<page | PROMPT>`]

Packs arbitrary files into a PDF using PDF's file attachment features. More than one attachment may be listed after `attach_files`. Attachments are added at the document level unless the optional `to_page` option is given, in which case the files are attached to the given page number (the first page is 1, the final page is `end`).

```
# Attach two XML files to page 11
pdftk in.pdf attach_files f1.xml f2.xml to_page 11 output out.pdf
```

**unpack\_files**

Copies all of the attachments from the input PDF file into the current directory or to an output



directory given after the output switch.

```
# Unpack attachments into directory "mypdfs"
pdftk in.pdf unpack_files output mypdfs/attachfiles
```

### **burst**

Splits a single, input PDF document into individual pages. Also creates a report named `doc_data.txt` which has the same format as the output from `dump_data`.

```
# Which pdf files are present in the current directory
> ls *.pdf
exapdflatex.pdf
# Split file exapdflatex.pdf into one-page fragments
> pdftk exapdflatex.pdf burst
# After the burst operation we find 12 pdf files
> ls *.pdf
exapdflatex.pdf  pg_0003.pdf  pg_0006.pdf  pg_0009.pdf
pg_0001.pdf      pg_0004.pdf  pg_0007.pdf  pg_0010.pdf
pg_0002.pdf      pg_0005.pdf  pg_0008.pdf  pg_0011.pdf
```

### **fill\_form <FDF data filename | - | PROMPT>**

Fills the form fields of the single input PDF file with data from an FDF file or `stdin` (see the program documentation for more details).

### **background <background PDF filename | - | PROMPT>**

Paints a PDF watermark as background on all pages of a single input PDF file. `pdftk` only uses the first page from the background PDF file and scales and rotates its contents as needed to fit the input page.

```
# Paint contents of image.pdf onto each page of file out.pdf
pdftk in.pdfbackground image.pdf output out.pdf
```

If the input PDF does not have a transparent background (e.g., a PDF created from page scans) then the resulting background will be invisible. In this case use the `stamp` feature instead.

### **stamp <stamp PDF filename | - | PROMPT>**

This behaves just like the `background` feature except it overlays the stamp PDF page on top of the input PDF document's pages. This works best if the stamp PDF page has a transparent background.

### **dump\_data**

Reports various statistics, metadata, bookmarks (i.e., “outlines”), and page labels as read in a single input PDF file. In the following example we display the bookmarks of the PDF documents we generated in Section 22.5.3. In particular, Figure 22.8 on page 156 shows clearly the correspondence of what is seen on screen and the bookmark information which is present in the file as printed by `pdftk`:

```
> pdftk exadvipdfmx.pdf dump_data
InfoKey: Creator
```

```

InfoValue: LaTeX with hyperref package
InfoKey: Producer
InfoValue: dvipdfmx (20050823)
InfoKey: CreationDate
InfoValue: D:20060507210911+00'00'
NumberOfPages: 11
BookmarkTitle: Introduction
BookmarkLevel: 1
BookmarkPageNumber: 1
BookmarkTitle: Landau theory
BookmarkLevel: 1
BookmarkPageNumber: 2
BookmarkTitle: Restrictions
BookmarkLevel: 2
BookmarkPageNumber: 3
BookmarkTitle: Vavilov theory
BookmarkLevel: 1
BookmarkPageNumber: 4
BookmarkTitle: Gaussian Theory
BookmarkLevel: 1
BookmarkPageNumber: 5
BookmarkTitle: Urb&#225;n model
BookmarkLevel: 1
BookmarkPageNumber: 5
BookmarkTitle: Fast simulation for n3 16
BookmarkLevel: 2
BookmarkPageNumber: 7
BookmarkTitle: Special sampling for lower part of the spectrum
BookmarkLevel: 2
BookmarkPageNumber: 9

> pdftk exadvips.pdf dump_data
InfoKey: Creator
InfoValue: LaTeX with hyperref package
InfoKey: Producer
InfoValue: dvips + GNU Ghostscript 7.05
Rest identical to previous case

> pdftk exapdflatex.pdf dump_data
InfoKey: Creator
InfoValue: LaTeX with hyperref package
InfoKey: Producer
InfoValue: pdfTeX-1.304
InfoKey: PTEX.Fullbanner
InfoValue: This is pdfTeX, Version 3.141592-1.30.4-2.2 (Web2C 7.5.5)
InfoKey: CreationDate
InfoValue: D:20060507205722+02'00'
PdfID0: d5383689d3b9b22a523bfd24963d5ff
PdfID1: d5383689d3b9b22a523bfd24963d5ff
Rest identical to previous case

```

### **dump\_data\_fields**

Reads a single, input PDF file and reports form field statistics.

**update\_info** <info data filename | - | PROMPT>

Changes the metadata stored in a single PDF's Info dictionary to match the input data file. The input data file uses the same syntax as the output from `dump_data`.

**[output** <output filename | - | PROMPT>]

Specifies the output PDF filename, which must be different from any input filename. Use - to output to stdout.

```
# Wildcards to concatenate all pdf files in the current directory
pdftk *.pdf cat output all.pdf
```

**[encrypt\_40bit | encrypt\_128bit]**

The encryption strength for a user or owner password to be assigned to an output document defaults to 128 bits. This can be overridden by specifying `encrypt_40bit`.

**[allow** <permissions>]

Permissions are applied to the output PDF only if an encryption strength is specified or an owner or user password is given. If permissions are not specified, they default to “none,” and all features are disabled (see the program documentation for more details).

**[owner\_pw** <owner password | PROMPT>]

**[user\_pw** <user password | PROMPT>]

If an encryption strength is given but no passwords are supplied, then the owner and user passwords remain empty, which means that the resulting PDF may be opened and its security parameters altered by anybody.

```
# encrypt PDF using 128-bit strength (default)
#           and withhold all permissions (default)
pdftk in.pdf output out128.pdf owner_pw Opass

# as above, but user needs password to open the PDF
pdftk in.pdf output out128.pdf owner_pw Opass user_pw Upass

# as above, but after the PDF file is open only printing is allowed
pdftk in.pdf output out128.pdf owner_pw Opass user_pw Upass allow printing
```

**[compress | uncompress]**

Remove PDF page stream compression by applying the uncompress filter so that the file can be changed or inspected in an editor, such as emacs. After editing the compress filter will restore compression.

```
# uncompress PDF file for editing the PDF code in a text editor
pdftk in.pdf output editable.pdf uncompress
```

**[flatten]**

This option merges interactive form fields (and their data) associated with an input PDF file with the pages of that file.

#### **[verbose]**

By default, `pdftk` runs quietly. The `verbose` option will provide more information about the run.

#### **[dont\_ask | do\_ask]**

`pdftk` may prompt for further input when it encounters a problem, such as a bad password. You can control the default behavior by specifying `do_ask` (`pdftk` will ask you what to do) or `dont_ask` (`pdftk` will *not* ask you what to do). In `dont_ask` mode, `pdftk` overwrites files with its output without prompting.

An interesting example of the use of the `background` option is coloring the background on each page of a PDF document, as follows:

```
echo "%BoundingBox: 0 0 595 842
0.9 0.9 0.8 setrgbcolor      % set color
% start drawing the rectangle
0 0 moveto 595 0 rlineto 0 842 rlineto -595 0 rlineto
% close it and fill it with the chosen color
closepath fill
showpage " |
ps2pdf - |
pdftk article.pdf background - output article_color1.pdf
```

For changing the color you can adjust the three values preceding the `setrgbcolor` command. You can also store the EPS part of the above in a file and use:

```
more co.eps | ps2pdf - | \
pdftk article.pdf background - output article_color2.pdf
```

When operating on a single PDF file, `pdftk` preserves its bookmarks. On the other hand, when combining PDF files or when extracting pages into a new PDF, the resulting PDF file has no bookmarks.

## 22.6.2 mbtPDFasm

Thierry Schmitt developed the program `mbtPDFasm`,<sup>1</sup> which can assemble or merge PDF files, and extract information from or update the metadata associated with PDF files. Compared to `pdftk` this program is very fast if you wish to simply assemble PDF files resulting from a scanner, or output by other software, etc. It also offers a simple and convenient way to update the global metadata associated to a PDF file.

The program features the following options.

```
mbtPdfAsm [-mM]Mask -dDest [-a] [-as] [-b] [-z] [-r]
          [-oFilename] [-pPageList] [-sFilename]
          [-c[RUOL]Param] [-g[HACFKNST]] [-u[PK]] [-lPx] [-S]
          [-tText] [-T[cfosxy]Value] [-n] [-N[0cfsxy]Value] [-RValue]
```

<sup>1</sup>Available from <http://thierry.schmit.free.fr/dev/mbtPdfAsm/enMbtPdfAsm2.html>, where you can also find a Java-based Graphic Interface for `mbtPdfAsm`.

- Mask** Perl-like regular expression identifying the files to be assembled. Several masks can be combined by separating them with a semicolon (;) or with a comma (,) on Linux. When using `-M` instead of `-m` the interpretation of the mask will be “regularized”, i.e., `*` stands for any number of arbitrary characters, and `?` for any *single* character. By default, files are sorted in alphabetic order before assembling.
- Dest** Name of the assembled result. If *Dest* comprises a directory “path”, the latter must already exist. When the destination file matches a mask, the destination file will be overwritten, but not assembled.

On page 159 we used `pdftk`’s `burst` option to separate a file into one-page segments. We can re-assemble them into a single PDF file `all.pdf` as follows:

```
> mbtPdfAsm -Mpg*pdf -dall.pdf
mbtPdfAsm version 1.0.25
./pg_0001.pdf inserted (1 page(s)).
./pg_0002.pdf inserted (1 page(s)).
./pg_0003.pdf inserted (1 page(s)).
./pg_0004.pdf inserted (1 page(s)).
./pg_0005.pdf inserted (1 page(s)).
./pg_0006.pdf inserted (1 page(s)).
./pg_0007.pdf inserted (1 page(s)).
./pg_0008.pdf inserted (1 page(s)).
./pg_0009.pdf inserted (1 page(s)).
./pg_0010.pdf inserted (1 page(s)).
./pg_0011.pdf inserted (1 page(s)).
end of process, 11 page(s) assembled
```

### 22.6.2.1 Program options

- a Assemble files without sorting them.
- as Assembled files in the order specified by the mask.
- b Base directory from which the selection mask is applied.
- c Acces restrictions:
  - R** define restrictions;
  - a** add annotations or forms;
  - m** modify;
  - p** print;
  - s** select and extract;
  - U** opening password;
  - O** modifies restriction password;
  - L** length of encryption key (5 means 40 bits and 16 means 128 bits, the only two values supported by Adobe Acrobat).

We can assemble the files we concatenated in the previous example and impose a password `my-pass` for reading the combined file `all1.pdf`, as follows.

```
> mbtPdfAsm -mpg.*\pdf -cRp -cOmypass -dall1.pdf
```

When specifying the selections `-cRamps` all listed restrictions will be applied. When merging, if the restrictions on an input file are more restrictive than those specified on the command line, the more restrictive restriction will take effect.

**-g** Get information from the file(s) matching the mask(s):

**A** author;  
**C** encryption information (handler, key length, user password);  
**H** header line for the displayed information;  
**F** filename;  
**K** keywords;  
**N** number of pages in the file;  
**S** subject;  
**T** title.

The information is printed on standard output and on separate lines. We can look what the file `exapdflatex.pdf` contains as metadata, and obtain the following:

```
> mbtPdfAsm -Mexapdflatex.pdf -gACHFKNST
file name;number of pages;Author;keywords;Subject;Title;
security handler;key length;user password;
./exapdflatex.pdf;11;;;;;0;0;134720045;
```

We see that not much metadata is present. Below we shall add some with the `-s` and `-u` options.

- l** Limit the number of pages in the resulting PDF file(s) (see the program documentation).
- n** Insert a number from 0 in the assembled page.
- N** Format of the numbering displayed with the `-n` option (see the program documentation).
- o** Name of the file containing the description of the outlines (bookmarks, see the program documentation for the description of the format of this file).
- oo** Keep outlines.
- p** List of pages to be extracted from the files matching the mask of the `-m/M` option (see the program documentation for a description of the syntax). For instance, the first page of all `.pdf` files in the current directory is extracted into the file `allpage1.pdf` with the command:

```
> mbtPdfAsm -M*.pdf -pl -dallpage1.pdf
```

- r** Look for files recursively.
- R** Rotate all the pages of the destination file (possible values are 90, 180, and 270).
- s** Name of a script file, whose name must not begin by `A`, `K`, `S`, or `T` (the program documentation contains a description of its format). If the update mode is set, this option can also be used to specify the metadata to be updated with the following convention: `-sA` identifies the “Author”, `-sK` the “Keywords”, `-sS` the “Subject”, and `-sT` the “Title” (in the specification spaces should be replaced by underscores). An example is given below.
- S** Run the application in silent mode, (no information is displayed).
- t** Text to be added at the bottom of the assembled pages.
- T** Format the text displayed with the `-t` option (see the program documentation).
- u** Set update mode.
  - P** erase the metadata data when no value is specified on the command line;
  - K** erase the original file, otherwise a `pdfbak` file is saved.
- z** Invert the alphabetic order of the files to be assembled.

We can introduce metadata into the file `exapdflatex.pdf` (which contained none, see the example

of the `-g` option above) with the `-u` (update) and `-s` options, plus restrict the resulting file to read access only (`-c` option), as follows:

```
> mbtPdfAsm -Mexapdflatex.pdf -u -cRp \
>           -sAMaria_Physicist -sKLaTeX_PDF_math \
>           -sSMaking_PDF -sTEnergy_loss_stragglng
file ./exapdflatex.pdf updated
```

Using the `-g` option we can check whether the metadata have been entered correctly.

```
> mbtPdfAsm -Mexapdflatex.pdf -gACHFKNST
file name;number of pages;Author;keywords;Subject;
      Title;security handler;key length;user password;
./exapdflatex.pdf;11;Maria Physicist;LaTeX PDF math;Making PDF;
      Energy loss stragglng;/Standard;40;0;134720045;
```

We can also obtain information about a PDF file with the `pdftinfo` program. We once more see that the metadata have been entered correctly into the file. We also notice the effect of the `-cRp` option: the “Encrypted” line shows that only the “print” entry is not encrypted (and hence allowed), while all other actions are disallowed.

```
> pdftinfo exapdflatex.pdf
Title:      Energy loss stragglng
Subject:    Making PDF
Keywords:   LaTeX PDF math
Author:     Maria Physicist
Creator:    LaTeX with hyperref package
Producer:   mbt PDF assembleur version 1.0.25
CreationDate: Sun May  7 20:57:22 2006
Tagged:     no
Pages:      11
Encrypted:  yes (print:no copy:yes change:yes addNotes:yes)
Page size:  595.276 x 841.89 pts (A4)
File size:  181503 bytes
Optimized:  no
PDF version: 1.4
```

### 22.6.3 Using java for handling PDF files

**Multivalent** (<http://multivalent.sourceforge.net/Tools/index.html>) is a java library of utilities for manipulating PDF files. The following functions are provided.

**Compress** The compression algorithm removes duplicated or unused objects, strips off regenerable objects (ASCII filters, thumbnails), and replaces LZW compression with the superior Flate (also on inline images), and performs other optimizations. Command line options make it possible to initiate possible further space savings.

**Uncompress** The output file will contain the contents of the input PDF file in an pretty-printed uncompressed format, that can be easily inspected and edited (e.g., fixing typos, adding keywords, title, and annotations, etc. Note, however, that text will not be reflowed). After editing, the file should be handled with the **Compress** tool to recompress the streams and rebuild the cross-reference table.

- Impose** The tool arranges one or more existing pages onto a single output page. Simple “n-up” (e.g., shrinking two or four pages onto one) as well as more complex layouts (booklets, rotated pages, folded brochures, etc.) are possible (compare this with the `psutils` tools described in Section 22.4.1).
- Info** Displays information about the PDF document specified, such as its title, author, page count, page size, metadata, annotations, use of fonts.
- Decrypt** When supplying the owner password this tool decrypts PDFs with standard security, 40-bit or 128-bit keys.
- Encrypt** Encrypts a PDF file using standard 40-bit or 128-bit encryption. Moreover, restrictions can be specified for document use, such as disallowing copy and paste or high quality printing.
- Split** Splits one PDF file into one or more new PDF documents. Subsets can be specified, pages can be deleted, rearranged, duplicated, or blank pages can be inserted.
- Merge** Merges two or more PDF files into one, preserving named destinations, outlines, and interactive forms. Naming conflicts of destinations are handled by renaming the second and further occurrences.
- Repair** Repairs PDF files that are not too severely damaged. When running into hard problems where it cannot fix the damage, the tool tries to read the PDF data objects sequentially, and will collect all intact objects. The user can then try and fix the problem using `Uncompress` to generate an editable file, followed by `Compress`.
- Validate** Examines PDF files. The desired level of detail and error reporting can be selected.
- Undo** Undoes the last incremental update in a PDF file (e.g., annotations added by Adobe Acrobat or with Multivalent’s `Merge` tool and its `-append` option can be undone that way).
- Extract** Extract text from a PDF document. The text can be normalized to Unicode. Options allow one to also extract document structure, hyperlinks, layout (bounding boxes), and style (fonts).

We can test this tool set with the file `exapdflatex.pdf` mentioned in Section 22.5.3.

```
> export CLASSPATH=path to Multivalent.jar
> java tool.pdf.Validate exapdflatex.pdf
> java tool.pdf.Uncompress -fonts exapdflatex.pdf
exapdflatex-u.pdf is UNCOMPRESSED and pretty-printed, and can be edited
> java tool.pdf.Info -all exapdflatex.pdf
Filename: exapdflatex.pdf
Title: Energy loss straggling
Author: Maria Physicist
Subject: Making PDF
Keywords: LaTeX PDF math
Creator: LaTeX with hyperref package
Producer: mbt PDF assembleur version 1.0.25
Created: Sun May 07 19:57:22 GMT+01:00 2006
Page count: 11
PDF version: 1.4
outline
Encrypted: filter Standard (revision 2), 40-bit key, null password
restrictions: print fill forms accessibility assemble
anno: page 1, object 52, Link, bounds 81.0x9.0@(123.0,547.0)
anno: page 1, object 53, Link, bounds 91.0x11.0@(123.0,524.0)
```



```

anno: page 1, object 54, Link, bounds 77.0x9.0@(138.0,514.0)
anno: page 1, object 55, Link, bounds 91.0x11.0@(123.0,490.0)
anno: page 1, object 56, Link, bounds 102.0x11.0@(123.0,468.0)
anno: page 1, object 57, Link, bounds 83.0x9.0@(123.0,448.0)
anno: page 1, object 58, Link, bounds 145.0x11.0@(138.0,434.0)
anno: page 1, object 59, Link, bounds 233.0x11.0@(138.0,422.0)
  Annotations on pages 2 to 11
FONT NAME                                TYPE          ENCODING      EMB SUB UNI  OBJ#
CMBX10                                  Type1         <custom>      Y  Y  N    22
CMBX12                                  Type1         <custom>      Y  Y  N    17
CMEX10                                  Type1         <custom>      Y  Y  N    74
CMMI5                                   Type1         <custom>      Y  Y  N   148
CMMI7                                   Type1         <custom>      Y  Y  N    47
CMMI10                                  Type1         <custom>      Y  Y  N    32
CMMI12                                  Type1         <custom>      Y  Y  N   153
  More font declarations
Helvetica-Bold                          Type1C        <intrinsic>   Y  Y  N   130
Symbol                                  Type1C        <intrinsic>   Y  Y  N    93
Symbol                                  Type1C        <intrinsic>   Y  Y  N    97
Symbol                                  Type1C        <custom>      Y  Y  N   134
297 objects

22 fonts: 22 Type 1 4 embedded [QRIQJA+CMR17, TEGQEH+CMR12,
YHUMEK+CMBX12, EYUMTP+CMBX10, JNXCTC+CMR10, OKRQCL+CMMI10,
NORONU+CMR7, UWSHVF+CMSY10, EBVKFV+CMMI7, USWTGA+CMTT10,
HUDSBO+CMEX10, WLAXES+CMSY7, TQTJOW+Helvetica-Bold,
KLKJDZ+Symbol, KLKJDZ+Symbol, EJCTEH+CMTI10, STIZIF+CMR5,
EXQQQB+Helvetica-Bold, VRHQFE+Symbol, XXVQWO+CMMI5,
HUCCJV+CMMI12, ORVDWN+CMR8]

4 forms
  54 annotations:   54 hyperlinks

(11 pages) Content streams command usage: w 49 J 49 d 49 q 53 Q 53 cm
106 m 49 l 49 S 49 BT 60 ET 60 Tf 1373 Td 1620 TJ 1620 Do 2

```

We first have to tell java where the class library resides (the CLASSPATH variable). We check that the file is correct (the `Validate` tool does not complain, so it is). Then we uncompress the PDF file (`Uncompress` tool) and obtain a pretty-printed `exapdflatex-u.pdf`, which we can browse, edit, etc. Next we run the `Info` tool. Its (partial) output shows the characteristics of the eight bookmarks (hyperlinks) on page 1 (see Figure 22.8 on page 156). We have omitted the many lines of information displayed by the `Info` tool about the hyperlinks present on the other pages of the document and have replaced it with the single line *Annotations on pages 2 to 11*. In fact, our PDF document contains a total of 54 hyperlinks. They were all created automatically by the `hyperref` package, which was loaded in the preamble of the  $\text{\LaTeX}$  file `exa.tex` (see Section 22.5.3), and which transforms all references to figures, tables, equations, index entries and bibliographic entries, etc. into hyperlinks. We also have a list of all fonts used in the document (sometimes a font name appears several times since the PDF generating applications can embed several subsets of the same font as separate objects). The total number of fonts (22), annotations (54), hyperlinks (54), and pages (11) is displayed, as well as the usage statistics of the PDF operators in the content streams.

```

> export CLASSPATH=path to Multivalent.jar
> java tool.doc.ExtractText exapdflatex.pdf

```

```

URI: file:/pathtofile/exapdflatex.pdf
producer: pdfTeX-1.304
creator: LaTeX with hyperref package
pages: 11

```

Simulation of Energy Loss Straggling

Maria Physicist

May 7, 2006

Contents

```

1 Introduction 1
2 Landau theory 2
2.1 Restrictions ..... 3
3 Vavilov theory 4
4 Gaussian Theory 5
5 Urbán model 5
5.1 Fast simulation for n
3
≥ 16 ..... 7
5.2 Special sampling for lower part of the spectrum....9
1 Introduction

```

Due to the statistical nature of ionisation energy loss, large fluctuations can occur in the amount of energy deposited by a particle traversing an absorber element. Continuous processes such as multiple scattering and energy loss play a relevant role in the longitudinal and lateral development of electromagnetic and hadronic showers, and in the case of sampling calorimeters the measured resolution can be significantly affected by such fluctuations in their active layers. The description of ionisation fluctuations is characterised by the significance parameter  $\kappa$ , which is proportional to the ratio of mean energy loss to the maximum allowed energy transfer in a single collision with an atomic electron

$\kappa =$

$\xi$

$E$

$\max$

$E$

$\max$

is the maximum transferable energy in a single collision with an atomic electron.

The Extract tool run on the first page of the file `exapdflatex.pdf` (see Figure 22.8 on page 156) does a fairly good job when extracting the text. In the output file input characters are substituted by their Unicode equivalent as far as possible (e.g., we see the Greek symbols in the formulae displayed correctly). Of course, the program cannot correctly show the mathematical layout of the formulae.

We can investigate the structure of the PDF file by specifying more options:

```

> export CLASSPATH=path to Multivalent.jar
> java tool.doc.ExtractText -output xml exapdflatex.pdf
1 documents, length = 181008 => 63844, in 1.661 sec
<?xml version='1.0' encoding='UTF-8'?>
<document>
<page>

```

```

<pdf>
  <mediabox>
    <crop>
      <text>
        <qriqjacmr17>Simulation of Energy Loss Straggling</qriqjacmr17>
        <tegqehcmr12>Maria Physicist May 7, 2006</tegqehcmr12>
        <yhumekcmbx12>Contents</yhumekcmbx12>
        <eyumtpcmbx10>1 Introduction 1
        2 Landau theory 2</eyumtpcmbx10>
        <jnxctccmr10>2.1 Restrictions ..... 3</jnxctccmr10>
        <eyumtpcmbx10>3 Vavilov theory 4
        4 Gaussian Theory 5
        5 Urban model 5</eyumtpcmbx10>
        <jnxctccmr10>5.1 Fast simulation for</jnxctccmr10>
        <okrqclcmml10>n</okrqclcmml10> <noronucmr7>3</noronucmr7>
        <uwshvfcmsy10>≥</uwshvfcmsy10> <jnxctccmr10>16....7
        5.2 Special sampling for lower part of the spectrum ... 9</jnxctccmr10>
        <yhumekcmbx12>1 Introduction</yhumekcmbx12>
        <jnxctccmr10>Due to the statistical nature of ionisation energy loss,
        large fluctuations can occur in the amount of energy deposited by a
        particle traversing an absorber element. Continuous processes such as
        multiple scattering and energy loss play a relevant role in the
        longitudinal and lateral development of electromagnetic and hadronic
        showers, and in the case of sampling calorimeters the measured
        resolution can be significantly affected by such fluctuations in
        their active layers. The description of ionisation fluctuations is
        characterised by the significance parameter</jnxctccmr10>
        <okrqclcmml10>κ</okrqclcmml10> <jnxctccmr10>, which is
        proportional to the ratio of mean energy loss to the maximum allowed
        energy transfer in a single collision with an atomic
        electron</jnxctccmr10>
        <okrqclcmml10>κ</okrqclcmml10> <jnxctccmr10>=</jnxctccmr10>
        <okrqclcmml10>ξ
      </okrqclcmml10></text>
      <okrqclcmml10><width><line />
      </width> </okrqclcmml10>
      <text><okrqclcmml10>E</okrqclcmml10>
      <noronucmr7>max</noronucmr7>
      <okrqclcmml10>E</okrqclcmml10>
      <noronucmr7>max</noronucmr7>
      <jnxctccmr10>is the maximum transferable energy in a single collision
      with an atomic electron.</jnxctccmr10>

```

With the help of an XML syntax, we discover the hierarchical tree structure of the document, including where a particular font is used. This presentation is very useful to understand the layout of a PDF file.

## 22.6.4 Handling PDF document with L<sup>A</sup>T<sub>E</sub>X

Andreas Matthias developed pdfpages, a package that lets you insert and manipulate pages of external PDF documents with pdf<sub>l</sub>atex, so that you can combine PDF documents generated by various applications, e.g., Adobe FrameMaker, Microsoft Office, Open Office, pdf<sub>l</sub>atex, and at the same time use L<sup>A</sup>T<sub>E</sub>X for generating title pages, front or back material, tables of contents, running titles, etc.

The pdfpages package can combine several logical pages on a single “sheet” and the layout of each

output page can be specified individually. A lot of hypertext operations are supported, like links to the inserted pages, links to the original PDF document, threads, etc.

The main command of the `pdfpages` package lets you include one or more PDF pages into the current document, as follows:

```
\includepdf[key/val-list]{file}
```

The `key/val-list` is a comma-separated list of `key=value` pairs for keys that take a value. For Boolean keys, specifying just the key is equivalent to `key=true`; not specifying the key is equivalent to `key=false`. The more important keys are listed below (see the package documentation for a complete list).

**pages=page-specs** (default: `pages=1`)

The selection of pages to insert is specified as a comma-separated list, containing individual page numbers (`pages={3, 6, 7, 9}`), page ranges (`pages={1-3, 6-11}`) or a combination of these. An empty page is inserted by specifying `{}`, e.g., `pages={1, 3, {}, 6-9, 13}` will insert pages 1, 3, an empty page, pages 6 to 9, and finally page 13. Special cases are `pages=-9`, which will include pages 1 to 9, `pages=9-`, which will include pages from 9 until the last page of the document, `pages=-` includes all pages in the input document.

**nup=nxxny** (default: `nup=1x1`)

Number of logical pages to typeset on each output sheet; `nx` and `ny` specify the number of logical pages in the horizontal and vertical directions, respectively.

**landscape[=true|false]** (default: `landscape=false`)

The orientation of the output sheet is “landscape” (i.e., rotated 90 degrees).<sup>1</sup>

**delta=x-delta y-delta** (default: `delta=0 0`)

Horizontal and vertical space to be added between logical pages (used in conjunction with `nup` parameter).

**offset=x-offset y-offset** (default: `offset=0 0`)

The two-dimensional shift to be applied to the origin of the inserted pages.

**frame[=true|false]** (default: `frame=false`)

A frame should be drawn around each logical page. The thickness of the frame is controlled by  $\TeX$ 's length parameter `\fboxrule`.

**pagecommand= $\TeX$  commands** (default: `pagecommand={\thispagestyle{empty}}`) List of  $\TeX$  commands to be executed on each output sheet.

**noautoscale[=true|false]** (default: `noautoscale=false`)

By default the `pdfpages` package scales pages automatically. The `noautoscale` key will suppress this behavior, thus allowing full control of the size of the output page by the use of the `scale` key, which works as described for the `graphicx` package.

**reflect** Generates mirror image of included pages.

**signature=size** Generate signatures for booklets. The `size` parameter specifies the size of the signature, which must be a multiple of 4. The `signature` must be combined with adequate settings for the `nup` key, e.g., `nup=1x2` or `nup=2x1`.

**addtotoc={ page, section, level, heading, label}**

An entry is added into the table of contents. The five arguments, which must be specified in the order shown, are the following:

**page** page number of the inserted page in the output document;

**section**  $\TeX$  sectioning command name (e.g., `section`, `subsection`);

**level**  $\TeX$  sectioning command (e.g., 1 for `section`, 2 for `subsection`);

<sup>1</sup>The input (logical) pages themselves are not rotated. To achieve rotation or scaling use the `rotate` or `scale` parameters of the `\includegraphics` command, which are also recognized by `pdfpages`.

**heading** text for the heading to be inserted in the table of contents;  
**label** key for the `\label` command that  $\TeX$  will associate with the entry, so that  $\TeX$ 's standard `\ref` and `\pageref` commands can refer to them.

**addtolist**=`{ page, type, heading, label }`

An entry is added into the list of figures, tables, etc. The four arguments, which must be specified in the order shown, are the following:

**page** page number of the inserted page in the output document;  
**type**  $\TeX$  name of the “floating” environment (e.g., `figure`, `table`);  
**heading** text for the entry to be inserted in the table of figures, etc.  
**label** key for the `\label` command that  $\TeX$  will associate with the entry, so that  $\TeX$ 's standard `\ref` and `\pageref` commands can refer to them.

Our first example extracts the first page of the PDF file `infile.pdf` and puts it on an A4 output page. In general it is convenient to use `pdfpages` together with the `geometry` package, which lets you precisely control the format of the output page, which in this case leaves the full area to be covered by the PDF input file. We generate a mirror image of the image, magnify it up by 9% and offset it towards the lower left corner by 5 mm in both directions.

```
\documentclass[a4paper]{report}
\usepackage{geometry}
\geometry{paperwidth=210mm,paperheight=297mm,
         width=210mm,height=297mm,
         right=0mm,bottom=0mm,left=0mm,top=0mm}
\usepackage{pdfpages}
\begin{document}
  \includepdf[pages=1,reflect,
             noautoscale,
             scale=1.09,
             offset=-5mm -5mm]
             {myfile.pdf}
\end{document}
```

Our second example rotates page 27 of the PDF file `infile.pdf` by 90 degrees and leaves the other pages untouched. Note the use of the page ranges for the `pages` key, as well as the `noautoscale` key to ensure that the size of the output pages is identical to that of the input pages.

```
\documentclass[a4paper]{report}
\usepackage{geometry}
\geometry{paperwidth=210mm,paperheight=297mm,
         width=210mm,height=297mm,
         right=0mm,bottom=0mm,left=0mm,top=0mm}
\usepackage{pdfpages}
\begin{document}
  \includepdf[pages=-26,noautoscale]{infile.pdf}
  \includepdf[pages=27,noautoscale,angle=90]{infile.pdf}
  \includepdf[pages=28-,noautoscale]{infile.pdf}
\end{document}
```

Next we want to trim the top 48 mm (the header) of the first page, since the first page of the output file will be printed on paper that contains already a pre-printed color variant of the header. We use the functions of the `graphicx` package to trim the material.<sup>1</sup> Notice also how we had to offset the page

<sup>1</sup>The `pdfpages` package loads the `graphicx` package.

towards the bottom (by half the amount cut off at the top), since the `pdfpages` package centers the material it inputs onto the output frame.

```
\documentclass[a4paper]{report}
\usepackage{geometry}
\geometry{paperwidth=210mm,paperheight=297mm,
         width=210mm,height=297mm,
         right=0mm,bottom=0mm,left=0mm,top=0mm}
\usepackage{pdfpages}
\begin{document}
  \includepdf[pages=1,
             trim=0mm 0mm 0mm 48mm,clip,
             noautoscale,offset=0mm -24mm]{infile.pdf}
  \includepdf[pages=2-,noautoscale]{infile.pdf}
\end{document}
```

If we want to save trees we can combine several input pages onto a single output sheet, as the following example shows. We put four pages in landscape mode on a landscape output sheet, and adjust the spacing between the logical pages by 4 mm and 6 mm in the horizontal and vertical directions, respectively (somewhat similar to what the `pstops` and `psnup` tools described in Section 22.4.1 can do with PostScript files).

```
\documentclass[a4paper]{report}
\usepackage{geometry}
\geometry{paperwidth=210mm,paperheight=297mm,
         width=210mm,height=297mm,
         right=0mm,bottom=0mm,left=0mm,top=0mm}
\usepackage{pdfpages}
\begin{document}
  \includepdf[nup=2x2,landscape,pages=--,delta=4mm 6mm]{infile.pdf}
\end{document}
```

If you want somewhat more control over the placement of the logical pages, e.g., you want to make a booklet of A6 pages, then you can use something like the following setup.

```
\documentclass[a4paper]{report}
\usepackage{geometry}
\geometry{paperwidth=210mm,paperheight=297mm,
         width=210mm,height=297mm,
         right=0mm,bottom=0mm,left=0mm,top=0mm}
\usepackage{pdfpages}
\begin{document}
  \includepdf[pages={1,3,5,7},landscape,nup=2x2,viewport=10 10 560
             410,clip,offset=2mm -3mm]{infile.pdf}
  \includepdf[pages={4,2,8,6},landscape,nup=2x2,viewport=10 10 560
             410,clip,offset=-2mm -3mm]{infile.pdf}
```

A viewport is specified for the input pages for clipping purposes. The resulting A6 pages are further positioned optimally by offsetting their lower left corner. The selected pages are drawn left to right, top to bottom on the first two frames of the output document.

### 22.6.4.1 A script interface to pdfpages

A user-friendly interface to `pdfpages`, which eliminates the need for the user to run `pdftex` explicitly, is provided by three *PDFjam* scripts written by David Firth and which are available from <http://www.warwick.ac.uk/go/pdfjam>.

- `pdfjoin` concatenates pages of multiple PDF files into a single file;
- `pdfnup` combines several logical PDF pages onto a single output page;
- `pdf90` rotates pages of one or more PDF files counterclockwise through 90 degrees.

These Unix scripts gather the information the user provides as parameters and runs `pdflatex` with `pdfpages` in the background. The `-help` option displays for each of them the available options. A few examples follow.

```
> pdfjoin infile1.pdf infile2.pdf infile3.pdf --fitpaper true
```

Concatenates the three input files, keeping the original sizes and orientation into the single file `infile3-joined.pdf`.

```
> pdfjoin infile1.pdf infile2.pdf infile3.pdf --fitpaper false \
> --paper a4paper --outfile ~/newfiles/biga4file.pdf
```

Concatenates the three input files, scale them to fit on A4, if needed, and store them in the single output file specified as argument of the `--outfile` option.

```
> pdfnup --pages 1,3,5,7 --orient landscape --nup 2x2 \
> --offset '2mm -3mm' --outfile new.pdf infile.pdf
```

Selects pages 1, 3, 5 and 7 of the input file `infile.pdf` to be output onto one output page in the file `new.pdf` in landscape orientation and offset with the specified amount.

```
> pdf90 infile1.pdf infile2.pdf
```

Creates two new files, `infile1-rotated.pdf` and `infile2-rotated.pdf` in the current directory, retaining original page sizes in the output file.

### 22.6.4.2 Using $\text{\LaTeX}$ and pdfpages for typesetting

We end this section with a more complex example, where we combine  $\text{\LaTeX}$ 's typesetting engine for preparing title pages, front and back matter, for managing the table of contents via its structural cross-reference tools, and for outputting running headers and footers. The external PDF pages that are collected into the output file can be produced by various tools, such as Microsoft Word, Adobe FrameMaker, or `latex` itself. When preparing these PDF files, one should, of course, try and guarantee some uniformity of fonts and layout (body text, section titles, figure and table captions, etc.).

```
1 \documentclass[11pt,twoside]{report}
2 \usepackage{ifthen}
3 \newcommand{\BLKP}{% add blank page when on even page
4   \ifthenelse{\isodd{\value{page}}}{\relax}{\mbox{} \thispagestyle{empty} \newpage}}
5 \usepackage{geometry}
6 \geometry{paperwidth=210mm,paperheight=297mm,
7   right=18mm,bottom=24mm,left=22mm,top=24mm,
8   height=249mm,width=170mm,headsep=4mm,footskip=10mm,
9   headheight=14pt}
10 \usepackage{fancyhdr}
11 \pagestyle{fancy}
12 \fancyhf{}
```

```

13 \renewcommand\headrulewidth{0pt}
14 \fancyfoot[C]{\thepage}
15 \fancyhead[RE]{\textsc{\ARTauthor}\hspace*{3mm}}
16 \fancyhead[LO]{\hspace*{3mm}\textsc{\ARTtitle}}
17 \newcommand{\ARTauthor}{~}
18 \newcommand{\ARTtitle}{~}
19 \usepackage{pdfpages}
20 \begin{document}
21 \thispagestyle{empty}
22 \begin{center}
23 \Huge Text on title page\\
24 \includegraphics[width=15cm]{image.pdf}\\[10mm]
25 \includegraphics[width=15cm]{image.jpg}
26 \end{center}
27 \clearpage
28 \BKLP
29 \pagestyle{plain}
30 \pagenumbering{roman}
31 \setcounter{page}{3}
32 \begin{center}
33 \bfseries\Large Preface\\[1cm]
34 \end{center}
35
36 Text of preface
37
38 ... OTHER FRONT MATTER ...
39 \clearpage
40 \BKLP
41 \providecommand{\Tita}{Title of first contribution}
42 \providecommand{\Auta}{A. Auth1}
43 \providecommand{\Refa}{aut1}
44 \providecommand{\Titb}{Title of second contribution}
45 \providecommand{\Autb}{A. Auth2}
46 \providecommand{\Refb}{aut2}
47 ...
48 \begin{center}
49 \bfseries\Large Contents\\[5mm]
50 \end{center}
51 \begin{flushleft}
52 \Tita\\
53 \quad\emph{\Auta}
54 \quad\dotfill~\pageref{S\Refa}\\[3mm]
55 \Titb\\
56 \quad\emph{\Autb}
57 \quad\dotfill~\pageref{S\Refb}\\[3mm]
58 ...
59 \end{flushleft}
60 \newpage
61 \BLKP
62 \pagestyle{fancy}
63 \setcounter{page}{1}
64 \pagenumbering{arabic}
65 \newcommand{\Includeart}[3]{%
66 \renewcommand{\ARTauthor}{~}
67 \renewcommand{\ARTtitle}{~}
68 \includepdf[pages=1,noautoscale,offset=0mm 0mm,
69 pagecommand={\pagestyle{fancy}},
70 addtotoc={1, section, 0, dummy, S#3},
71 trim=19mm 21mm 19mm 27mm, clip]
72 {pdf/#3.pdf}
73 \renewcommand{\ARTauthor}{#2}
74 \renewcommand{\ARTtitle}{#1}
75 \includepdf[pages=2-,noautoscale,offset=0mm 0mm,
76 pagecommand={\pagestyle{fancy}},
77 trim=19mm 21mm 19mm 27mm, clip]
78 {pdf/#3.pdf}
79 \BLKP}
80 \Includeart{\Tita}{\Auta}{\Refa}
81 \Includeart{\Titb}{\Autb}{\Refb}
82 ...
83 ... BACK MATTER ...
84 \end{document}

```



After loading the `ifthen` package (line 2), we define a command `\BKLP` that will eliminate the running titles on a empty left-hand output page (lines 3–4). With the `geometry` package we define the visual layout of the page (lines 5–9), while the running headers and footers are defined with the `fancyhdr` package (lines 10–16). In particular, we want the page number centered at the bottom of each page (line 14), the author of the article at the top of even (left-hand) pages (line 15), and the title of the article of the top of odd (right-hand) pages (line 16); both author and title are initialized to “blank” (lines 17–18). Finally we load the `pdfpages` package (line 19).

The title page is typeset by  $\text{\LaTeX}$  and is followed by a blank page (lines 21–28). We initialize the page number to roman lowercase numbers starting at 3, and continue with typesetting the front matter (Abstract, Preface, etc., lines 29–40). As we do not use  $\text{\LaTeX}$  to typeset the body of the document, we have to specify the metadata (title, authors, reference key) for each contribution (article) to typeset the table of contents and control the output generated for the running titles (lines 41–47). With these definitions and the reference keys we construct the table of contents (lines 48–61).

Now we enter the main body of the document, for which we adopt the `fancy` style for the running titles and reset the page number and style (lines 62–64). The reference strings for author and title are put equal to a blank (line 66–67). To handle each contribution, we define the command `\Includeart` (lines 65–79), which starts by reading the first page of the PDF file (lines 68–72), which defines the reference for the page number in the table of contents (via the final argument of the `addtotoc` key on line 70) and outputs the page with an empty running header (lines 66–67). Then the remaining pages of the PDF file are read (lines 75–78). The settings for author and title are loaded (lines 73–74) so that they can be used for the running headers. Starting from line 80 we treat each contribution in turn, specifying its title (`\Tita`, etc.), author (`\Autha`, etc.), and filename (`\Ref a`, etc.)<sup>1</sup> This information was defined previously (lines 41–47).

Finally (line 83), if desired, we can conclude the document with some back matter (acknowledgments, photos, etc.).

### 22.6.5 Flipping PDF pages

In Section 22.2.6 we explained how one can “flip” (i.e., create a mirror image of) PostScript pages, as is sometimes required by publishers who want “camera-ready” documents on transparent films. In the case of PDF files, Section 22.6.4 describes the `reflect` option of Andreas Matthias’s `pdfpages` package. Building on his `everypage` package, Sergio Callegari developed the `pdfflip` package which can mirror one or more pages in a PDF file. By default, when loading the package all pages will be flipped. When specifying the `off` option:

```
\usepackage[off]{pdfflip}
```

the package is loaded but remains inactive. To start page flipping issue the command `\FlipPDF`, to turn flipping off, issue `\UnFlipPDF`.

### 22.6.6 The Glyph and Cog tools

The Company *Glyph & Cog, LLC* (<http://www.glyphandcog.com>) designs and implements software for manipulating electronic documents, including software libraries, components, and consulting services related to reading, viewing, and converting PDF files.

<sup>1</sup>The string that serves as a key to define the first page of each contribution so that it can be entered in the table of contents is also used as filename to make file management easier. For instance, the value of `\Ref a`, defined on line 43, is used when executing the command `\Includeart` on line 80, as a filename (lines 72 and 78) and as the key for defining the page reference (line 70).

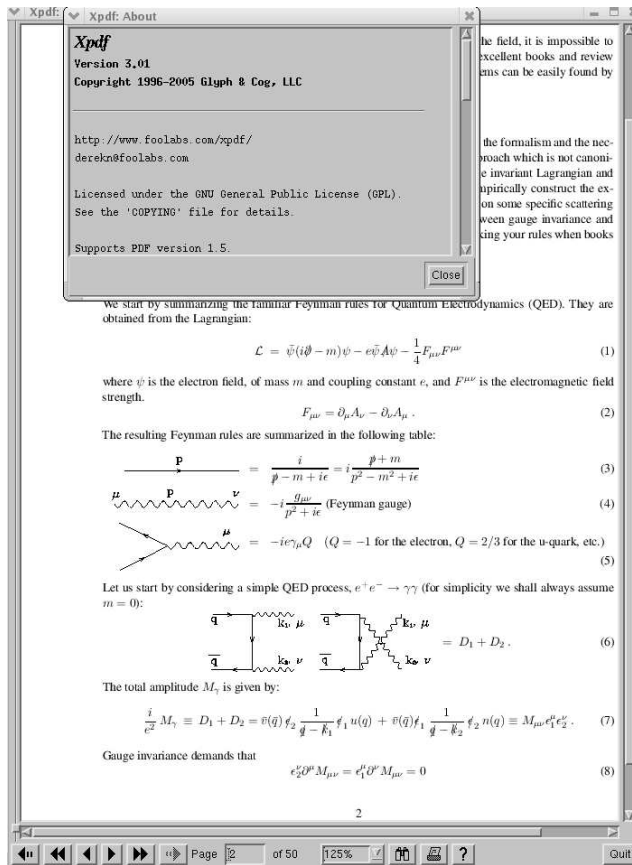


Figure 22.9: Example of the use of xpdf. The PDF version of page 2 of the same document that is displayed in Figures 22.3 and 22.4 on page 130 is shown.

Glyph & Cog also owns and maintains the open source Xpdf project (<http://www.foolabs.com/xpdf/>), which provides xpdf, a free PDF viewer (for X Windows on Unix) and a few associated command line utilities. The xpdf viewer (Figure 22.6.6) can use PostScript Type 1 and TrueType fonts, but, compared to the evince and ghostview viewers described in Section 22.3.4, only handles PDF files. The xpdf distribution does not contain any font files. As PDF files can refer to the “Base-14” fonts (*Times*, *Helvetica*, *Courier*, all three in regular, bold, italic, and bold-italic variants, *Symbol*, and *Zapf Dingbats*) without embedding them, the user should tell xpdf where to find PostScript Type 1 versions of these fonts. In principle the Ghostscript free, high-quality PostScript Type 1 fonts, can be used. The location of these fonts on your system should be specified in the `.xpdfrc` file in the user’s home directory (typing `man xpdfrc` will give you the necessary details). In this file you can also specify the print command and output page size if you want to print directly from xpdf.

The xpdf distribution comes with a series of command line programs that perform various tasks on PDF files. All these commands recognize the following common options:

- f integer** first page of PDF file to handle.
- l integer** last page of PDF file to handle.

- opw *pwstring*** owner password (for encrypted PDF file).
- upw *pwstring*** user password (for encrypted PDF file).
- v** display copyright and version information.
- h, -help, -help or -?** display usage information.

In general for unspecified options the defaults in `.xpdfrc` will be used. An alternative file configuration file can be specified with the `-cfg filename` option.

### 22.6.6.1 The pdftops program

The `pdftops` program generates a PostScript file from a PDF source. The simplest usage instance is:

```
pdftops file.pdf
```

This will generate `file.ps`. The output filename can be controlled by specifying a second argument, e.g.,

```
pdftops infile.pdf outfile.ps
```

will generate `outfile.ps`. An EPS instance of page two of the multi-page document `mangano.pdf` shown in Figure 22.6.6 on the facing page is created as follows:

```
pdftops -eps -f 2 -l 2 mangano.pdf manganop2.eps
```

Further options exist for controlling the level of PostScript code that is generated, the font embedding, and the paper size. A one-line summary of all available options is displayed by `pdftopdf -help`, while `man pdftops` provides a more detailed overview.

### 22.6.6.2 The pdftotext program

A plain text version of a PDF file can be generated by the `pdftotext` program. An example with the PDF file `exapdflatex.pdf`, discussed in Section 22.5.3. Compare this output with the first page displayed in Figure 22.8, and with the text output generated with the multivalent java tool discussed in Section 22.6.3. The `-layout` option tries to maintain the original physical layout.

```
> pdftotext -layout -l 1 -f 1 exapdflatex.pdf p1.txt
```

The contents of the output file `p1.txt` is as follows.

```

Simulation of Energy Loss Straggling
Maria Physicist
May 7, 2006

Contents
1 Introduction 1
2 Landau theory 2
  2.1 Restrictions . . . . . 3
3 Vavilov theory 4
4 Gaussian Theory 5
5 Urb'n model 5
  a 5
  5.1 Fast simulation for n3 16 . . . . . 7
```

5.2 Special sampling for lower part of the spectrum . . . . . 9

### 1 Introduction

Due to the statistical nature of ionisation energy loss, large fluctuations can occur in the amount of energy deposited by a particle traversing an absorber element. Continuous processes such as multiple scattering and energy loss play a relevant role in the longitudinal and lateral development of electromagnetic and hadronic showers, and in the case of sampling calorimeters the measured resolution can be significantly affected by such fluctuations in their active layers. The description of ionisation fluctuations is characterised by the significance parameter, which is proportional to the ratio of mean energy loss to the maximum allowed energy transfer in a single collision with an atomic electron

=

$E_{\max}$

$E_{\max}$  is the maximum transferable energy in a single collision with an atomic electron.

$$E_{\max} = \frac{2m_e c^2 \beta^2 \gamma^2}{1 + 2m_e \beta^2 \gamma^2 / m_x + (m_e \beta^2 \gamma^2 / m_x)^2},$$

where  $\beta = E/m_x$ ,  $E$  is energy and  $m_x$  the mass of the incident particle,  $\gamma^2 = 1 - 1/\beta^2$  and  $m_e$  is the electron mass. comes from the Rutherford scattering cross section and is defined as:

$$\sigma = \frac{2Z^2 e^4 N_A v}{m_e c^2 A} = 153.4 \frac{Z^2}{A} \times \text{keV},$$

1

### 22.6.6.3 The pdfinfo program

Information about a PDF file is displayed with the `pdfinfo` command, which we already encountered when describing `mbtPDFasm` in Section 22.6.2. Document-level metadata is displayed when specifying the `-meta` option, while the `-box` option prints the page bounding boxes, as the following example shows (compare this with the output on page 165).

```
> pdfinfo -f 1 -l 1 -box exapdflatex.pdf
Title:      Energy loss straggling
Subject:     Making PDF
Keywords:    LaTeX PDF math
Author:      Maria Physicist
Creator:     LaTeX with hyperref package
Producer:    mbt PDF assembleur version 1.0.25
CreationDate: Sun May  7 20:57:22 2006
Tagged:      no
Pages:       11
Encrypted:   yes (print:no copy:yes change:yes addNotes:yes)
Page   1 size: 595.276 x 841.89 pts (A4)
Page   1 MediaBox:   0.00   0.00  595.28  841.89
Page   1 CropBox:    0.00   0.00  595.28  841.89
Page   1 BleedBox:    0.00   0.00  595.28  841.89
Page   1 TrimBox:     0.00   0.00  595.28  841.89
Page   1 ArtBox:      0.00   0.00  595.28  841.89
File size:    181503 bytes
Optimized:    no
PDF version:  1.4
```

### 22.6.6.4 The pdffonts program

Information about each font used in a PDF file can be obtained with the `pdffonts` command. The information obtained for the fonts is somewhat similar to what we got with the multivalent tool and its (Section 22.6.3).

```
> pdffonts exapdflatex.pdf
```

name	type	emb	sub	uni	object	ID
QRIQJA+CMR17	Type 1	yes	yes	no	7	0
TEGQEH+CMR12	Type 1	yes	yes	no	12	0
YHUMEK+CMBX12	Type 1	yes	yes	no	17	0
EYUMTP+CMBX10	Type 1	yes	yes	no	22	0
JNXCTC+CMR10	Type 1	yes	yes	no	27	0
OKRQCL+CMMI10	Type 1	yes	yes	no	32	0
NORONU+CMR7	Type 1	yes	yes	no	37	0
UWSHVF+CMSY10	Type 1	yes	yes	no	42	0
EBVKFV+CMMI7	Type 1	yes	yes	no	47	0
USWTGA+CMTT10	Type 1	yes	yes	no	63	0
HUDSBO+CMEX10	Type 1	yes	yes	no	74	0
WLAXES+CMSY7	Type 1	yes	yes	no	79	0
TQTJOW+Helvetica-Bold	Type 1C	yes	yes	no	89	0
KLKJDZ+Symbol	Type 1C	yes	yes	no	93	0
KLKJDZ+Symbol	Type 1C	yes	yes	no	97	0
EJCTEH+CMTI10	Type 1	yes	yes	no	110	0
STIZIF+CMR5	Type 1	yes	yes	no	115	0
EXQQQB+Helvetica-Bold	Type 1C	yes	yes	no	130	0
VRHQFE+Symbol	Type 1C	yes	yes	no	134	0
XXVQWO+CMMI5	Type 1	yes	yes	no	148	0
HUCCJV+CMMI12	Type 1	yes	yes	no	153	0
ORVDWN+CMR8	Type 1	yes	yes	no	158	0

This is identical to the font information obtained via the multivalent tool (see also page 166)

```
> java tool.pdf.Info -fonts exapdflatex.pdf
```

```
Filename: exapdflatex.pdf
```

FONT NAME	TYPE	ENCODING	EMB	SUB	UNI	OBJ#
CMBX10	Type1	<custom>	Y	Y	N	22
CMBX12	Type1	<custom>	Y	Y	N	17
CMEX10	Type1	<custom>	Y	Y	N	74
CMMI5	Type1	<custom>	Y	Y	N	148
CMMI7	Type1	<custom>	Y	Y	N	47
CMMI10	Type1	<custom>	Y	Y	N	32
CMMI12	Type1	<custom>	Y	Y	N	153
CMR5	Type1	<custom>	Y	Y	N	115
CMR7	Type1	<custom>	Y	Y	N	37
CMR8	Type1	<custom>	Y	Y	N	158
CMR10	Type1	<custom>	Y	Y	N	27
CMR12	Type1	<custom>	Y	Y	N	12
CMR17	Type1	<custom>	Y	Y	N	7
CMSY7	Type1	<custom>	Y	Y	N	79
CMSY10	Type1	<custom>	Y	Y	N	42
CMTI10	Type1	<custom>	Y	Y	N	110
CMTT10	Type1	<custom>	Y	Y	N	63
Helvetica-Bold	Type1C	<intrinsic>	Y	Y	N	89

Helvetica-Bold	Type1C	<intrinsic>	Y	Y	N	130
Symbol	Type1C	<intrinsic>	Y	Y	N	93
Symbol	Type1C	<intrinsic>	Y	Y	N	97
Symbol	Type1C	<custom>	Y	Y	N	134

With `pdffonts` you can limit your query to one page, e.g., page 6, (see Figure 22.6 on page 141 showing the first nine pages of the document), which contains a picture which refers to non-TeX fonts (*Helvetica-Bold* and *Symbol*),.

```
> pdffonts -f 6 -l 6 exapdfplatex.pdf
name                               type          emb sub uni object ID
-----
JNXCTC+CMR10                       Type 1         yes yes no      27  0
USWTGA+CMTT10                      Type 1         yes yes no      63  0
OKRQCL+CMMI10                     Type 1         yes yes no      32  0
NORONU+CMR7                       Type 1         yes yes no      37  0
EBVKFV+CMMI7                      Type 1         yes yes no      47  0
UWSHVF+CMSY10                    Type 1         yes yes no      42  0
HUDSBO+CMEX10                    Type 1         yes yes no      74  0
STIZIF+CMR5                      Type 1         yes yes no     115  0
EXQQQB+Helvetica-Bold             Type 1C        yes yes no     130  0
VRHQFE+Symbol                    Type 1C        yes yes no     134  0
```

### 22.6.6.5 The `pdftosrc` program

Based on the `xpdf` paradigm Hàn Thế Thành wrote the `pdftosrc` program, which extracts a source file or a stream from a PDF file.

```
pdftoscr PDFfile [ stream-object-number ]
```

If both *PDFfile* and *stream-object-number* are present, `pdftosrc` extracts and uncompresses the PDF stream of the object specified and writes it to a file named `PDF-file.stream-object-number` stripping the file extension (e.g., `.pdf`) from the original filename. Existing files will be overwritten.

If only *PDFfile* is specified, `pdftosrc` extracts the embedded source file from the first found stream object with `/Type /SourceFile` and writes it to a file with the name `/SourceName` as defined in that PDF stream object. As an example of this case consider the following file, `pdftosrctest.tex`.

```
\documentclass{article}
\begin{document}
This small text file shows how to include
the source of the file inside the PDF output.
\immediate\pdfobj
stream attr {/Type /SourceFile /SourceName (\jobname.tex)}
file{\jobname.tex}
\pdfcatalog{/SourceObject \the\pdflastobj\space 0 R}
\end{document}
```

The command syntax used above is suited for including the PDF source file with `pdftex`, as shown next.

```
> cp pdftosrctest.tex save.tex
> ls -l pdftosrctest.tex save.tex
-rw-rw-r-- 1 goossens goossens 293 Nov  4 15:35 pdftosrctest.tex
-rw-rw-r-- 1 goossens goossens 293 Nov  4 15:51 save.tex
```

```

> pdflatex pdftoscrtext.tex
This is pdfTeX, Version 3.141592-1.40.3 (Web2C 7.5.6)
  %&-line parsing enabled.
entering extended mode
(./pdftoscrtext.tex
LaTeX2e <2005/12/01>
Babel <v3.8h> and hyphenation patterns for english, ... loaded
(/texlive/2007/texmf-dist/tex/latex/base/article.cls
Document Class: article 2005/09/16 v1.4f Standard LaTeX document class
(/texlive/2007/texmf-dist/tex/latex/base/size10.clo))
(./pdftoscrtext.aux)<<pdftoscrtext.tex>>
[1/texlive/2007/texmf-var/fonts/map/pdftex/updmap/pdftex.map] (./pdftoscrtext.aux) )
</texlive/2007/texmf-dist/fonts/type1/bluesky/cm/cmr10.pfb>
Output written on pdftoscrtext.pdf (1 page, 9082 bytes).
Transcript written on pdftoscrtext.log.
> pdftosrc pdftoscrtext.pdf
pdftosrc version 3.01
Source file extracted to pdftoscrtext.tex
> ls -l pdftoscrtext.tex save.tex
-rw-rw-r-- 1 goossens goossens 293 Nov  4 15:53 pdftoscrtext.tex
-rw-rw-r-- 1 goossens goossens 293 Nov  4 15:51 save.tex
> diff -s pdftoscrtext.tex save.tex
Files pdftoscrtext.tex and save.tex are identical

```

In the previous sequence of commands we first make a copy of the input file `pdftoscrtext.tex` onto `save.tex`. Next we run the file `pdftoscrtext.tex` through `pdflatex`, which generates the output file `pdftoscrtext.pdf`, containing its source `pdftoscrtext.tex` as an PDF object. Now we run `pdftosrc` which creates `pdftoscrtext.tex`, overwriting the original file. To check that the extracted source and the original are the same we listed the file characteristics before and after running the programs. Finally, we make a check for differences and the `diff` utility confirms that there are none.

When the object or source cannot be found `pdftosrc` issues an error message (see the manual for more details).

### 22.6.6.6 Tools for handling images

`pdftoppm` converts a PDF file to a series of bitmaps (one for each page). By default color portable pixmaps (PPM) are obtained. Specifying the `-gray` option will generate portable graymaps (PGM), while with the `-mono` option monochrome portable bitmaps (PBM) will be created. The resolution can be specified with the option `-r number` (default 150 dpi).

`pdftopbm` lets you directly generate monochrome images. This program also takes the `-r` option.

`pdftimages` reads pages of a PDF file, extracts the raw images (ignoring possible additional transformations present in the PDF) and writes them as color portable pixmaps (PPM), monochrome portable bitmaps (PBM), or, when the `-j` option is specified, as JPEG files (this only works for images encoded in DCT format, the others will be saved as PPM or PBM).

## 22.7 Color in the printing industry and separation

The printing industry generally uses either the CMYK color model for full-color printing and spot colors for one or two-color work. The latter are normally specified according to the Munsell and Pantone charts; more recently, the Focoltone and Trumatch systems have become common for color matching.

PostScript is almost universally used for high quality typeset output, and Level 2 of the PostScript language offers full support for color, with not only the simple RGB, CMYK, and HSB models, but also the CIE system and various special color spaces. The details of these systems, and algorithms for converting between color models, are discussed exhaustively in [2], pp. 210–248. There are very useful discussions of color in PostScript in [12] and [13]; [4] also has useful material. If you read the older books, you should note that full Level 2 PostScript provides a number of important new commands that considerably ease preparation of color separations.

Issues encountered when working with T<sub>E</sub>X in the professional printing industry are discussed in a useful article by Michael Sofka [17]. [5] also treats color separation from a T<sub>E</sub>X perspective, while [18] describes how to do separations at the level of bitmap output from T<sub>E</sub>X.

The T<sub>E</sub>X user can be largely shielded from the complications of the actual printing. A document containing color material can be typeset and run through a driver like dvips to create a color PostScript document that can be previewed on screen or printed on a color printer. The main problem is that the color rendering and precision of the common inkjet printers is not good enough to prepare copy for professional printing; however, they are very well suited to transparencies, handouts, posters etc., and it is well worth the effort to develop your use of T<sub>E</sub>X color for this type of work.

## 22.7.1 Color separation

To produce a “real” book or journal using offset printing, the printer will require four versions of each page that contain, respectively, the gray levels corresponding to the proportions of cyan, magenta, yellow, and black. Each page is overprinted four times with each of the colors. Figure 22.10, containing the five Olympic rings at the top and a multi-colored ellipse at the bottom, shows how adding the various color inks gives a colored picture its final form. We started by applying the cyan ink (top left), and then added the magenta (top right), yellow (bottom left), and finally the black inks (bottom right) to obtain the picture in full color. The four separate stages of the process, and the cumulative effect, are shown.

Color work is usually typeset on special film and to high tolerances, since each page is overprinted four times and registration must be exact. Some typesetting systems can produce the four separations automatically, but more commonly this is done by PostScript manipulation. Professional-quality typesetting packages and sophisticated graphics manipulation packages like Adobe Illustrator, Corel Draw and Adobe Photoshop handle color separation directly, while Adobe’s Separator program can work on arbitrary color PostScript. The effectiveness of programs like this depends on how color is specified in the PostScript file produced, and PostScript Level 2 provides a great deal of support for this.

When preparing material for separation, the user needs to understand some specific issues:

- custom or “spot” colors;
- knockout and overprint (see Section 22.7.3);
- trapping.

Print jobs commonly differentiate between work that can be handled with the CMYK model (process color) and work specifying a precise color (custom color). While process color pages are printed four times with proportions of the appropriate color, pages with a custom color are printed separately, with the ink specifically requested (commonly from standard sets like Pantone). The T<sub>E</sub>X color package does not directly support spot color.

Let us consider the simple example of a drawing of a yellow circle on a blue background (like in Fig. 22.11). It must be decided at some point whether to create this by getting the right values of C, M, Y and K<sup>1</sup> and then printing them superimposed on one another (this is called overprinting), or by

<sup>1</sup>The normal printing order is in fact first yellow, then cyan, magenta, and finally black.



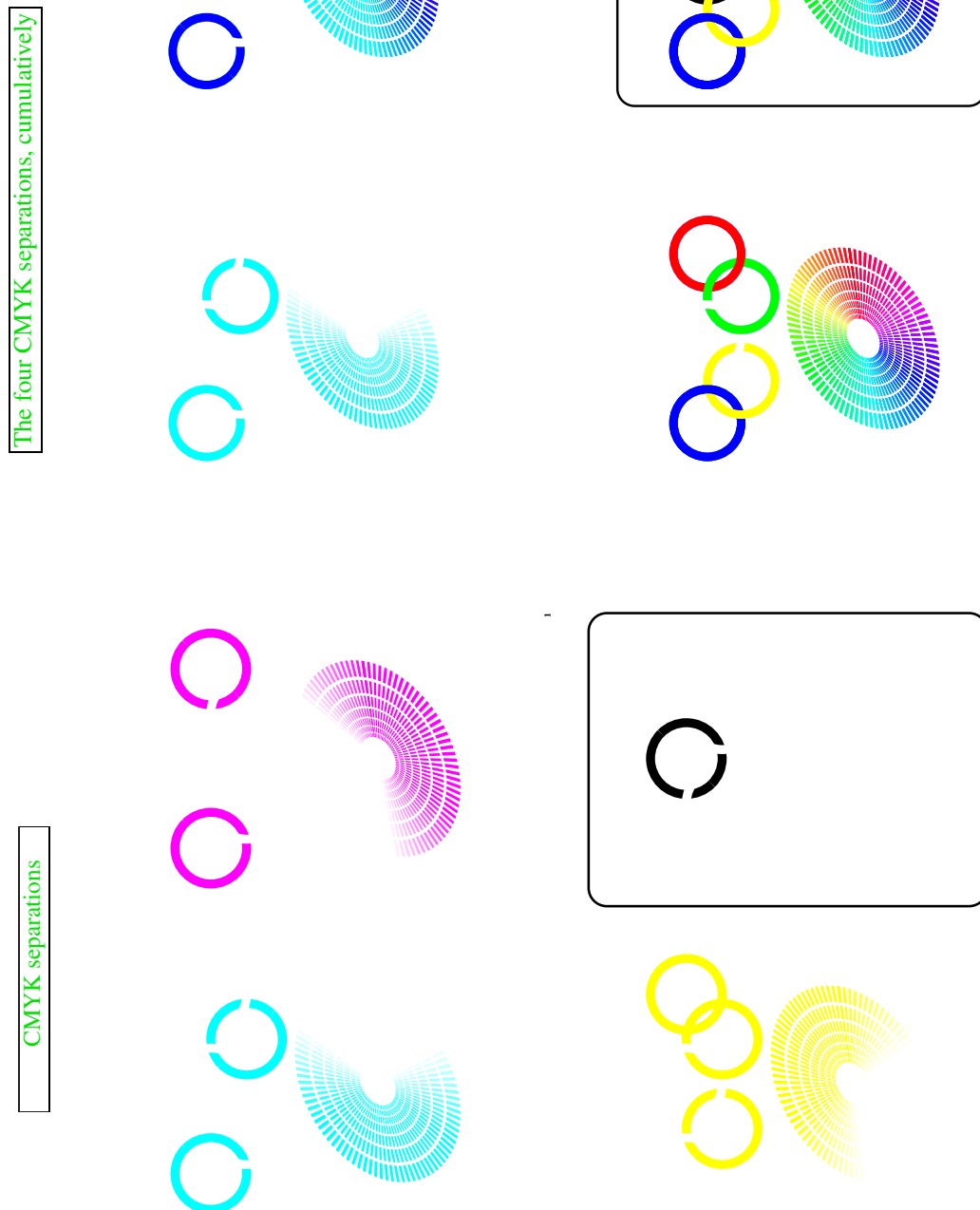


Figure 22.10: The separation of colors in the CMYK model.

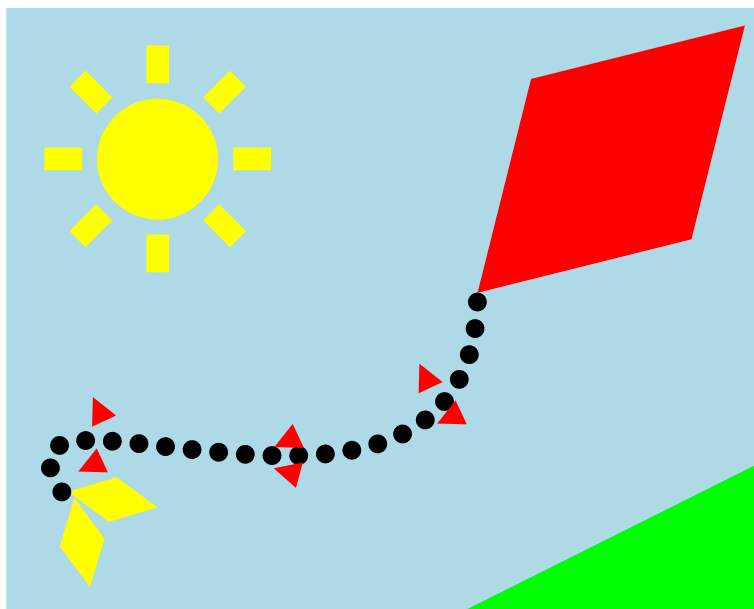


Figure 22.11: Simple PSTricks color example

leaving a circular hole in the blue background where the yellow circle will go (this is knockout). The CMYK system (and the default PostScript color model) assumes overprinting (since the whole system works on an additive principle), but custom colors naturally require a knockout, since they specify a precise color that should not be mixed with anything else. In PostScript, the `setoverprint` operator can be used to turn overprinting on or off; this could be used in more sophisticated color handling in a dvi-to-PostScript driver.

Whether we use knockout or overprinting, we will encounter another problem, registration of successive printings on a piece of paper. Even the most precise machinery has difficulty in aligning material that is specified in scaled points at a resolution of over 2500 dpi. In practice, therefore, careful color work uses *trapping*. Returning to the example of a yellow circle on a blue background with trapping, we either slightly decrease the size of the hole in the blue (a choke trap) or increase the size of the circle (a spread trap), and thus create a slight overlap to avoid registration problems.

It is beyond the scope of this book to deal with the full ramifications of high-quality color printing (such as the specification of screens), and we strongly suggest that readers take professional advice before embarking on large-scale full color projects with  $\text{\LaTeX}$ . While Mike Sofka ([17]) with a private dvi driver, and others using Textures on a Macintosh, have shown that  $\text{\LaTeX}$  can do full-scale color work, users who need relatively simple color printing (such as a two-color book in which all headings are blue and all tables have a blue tinted background) can do very well without recourse to commercial software. CMYK color separations can be made with the color package and dvips by using only PostScript Level 1 operators; we describe this in the following section.

### 22.7.2 Color separation using $\text{\LaTeX}$ and dvips

The principle of attaining simple dvips separations from  $\text{\TeX}$  output is that each output page is produced four times, using the `-b 4` command-line switch, or `b 4` in a configuration file, and a header file that

redefines the color operators for each of the four pages. The header file `colorsep.pro` (distributed with `dvips`, maintained by Sebastian Rahtz, and largely derived from [12] and [4]; see Section 22.7.2.1) uses the `bop-hook` handle (see Section 22.2.6 on page 111) to increment a counter at the beginning of each page, thus checking whether a C, M, Y or K page is being produced. The `setcmykcolor` operator is then redefined to produce just one of the four colors, in gray, and RGB colors are converted to CMYK before going through the same process. The `setgray` operator is activated only on the black (K) page. Unfortunately, color bitmap images are not handled by this system.

The output from the separation for the example of **Warning!** is simulated in Figure 22.12; here the box is set in “ForestGreen”, whose CMYK value is “0.91 0 0.88 0.12”. Notice that the “M” page is blank, as neither the green box nor the black text needs any magenta.

### 22.7.2.1 The `dvips` color separation header file

This is the contents of the file `colorsep.pro`, distributed with `dvips`, which redefines PostScript color operators to generate CMYK separations. It starts by defining some useful commands:

```

1  %!
2  /seppages 0 def
3  userdict begin
4  /Min {% 3 items on stack, find the smallest
5  2 copy lt { pop }{ exch pop } ifelse
6  2 copy lt { pop }{ exch pop } ifelse
7  } def

```

We need to convert color specified in the gray, HSB or RGB modes to CMYK for the four separations<sup>1</sup>. Gray is straightforward; HSB is simply a variant way of expressing RGB:

```

8  /SetGray {
9  1 exch sub systemdict begin adjustdot setgray end
10 } def
11 /sethsbcolor {systemdict begin
12   sethsbcolor currentrgbcolor end
13   userdict begin setrgbcolor end}def

```

RGB itself is harder (see [2, Section 7.2.3] on how to convert RGB to CMYK). In particular, we subtract the amount of each color from 1, and find the minimum black:

```

14 /ToCMYK
15 {
16   3 { 1 exch sub 3 1 roll } repeat
17   3 copy Min
18   blackUCR sub
19   dup 0 lt {pop 0} if
20   /percent_UCR exch def

```

Now we subtract that black undercolor from each color and work out the black itself:

```

21   3 { percent_UCR sub 3 1 roll } repeat
22   percent_UCR 1.25 mul % 1 exch sub
23 } def

```

<sup>1</sup>For conversions among the various device color spaces known to PostScript, see [2, Section 7.2].

The PostScript stack should now have the C, M, Y, and K values.

The CM command prints cropmarks:

```

24 /cX 18 def
25 /CM{gsave TR 0 cX neg moveto 0 cX lineto stroke
26 cX neg 0 moveto cX 0 lineto stroke grestore}def

```

The critical part is the definition of `bop-hook`, which is executed at the start of each page; this goes around a cycle of four pages, doing a different separation on each. Below we assume that each page is duplicated four times using the `-b 4` option of `dvips`:

```

27 /bop-hook{cX dup TR
28   seppages 1 add
29   /seppages exch def
30   seppages 5 eq { /seppages 1 def } if
31   seppages 1 eq {
32     /ColorName (CYAN) def
33     CYAN setupcolor
34     /WhichColor 3 def } if
35   seppages 2 eq {
36     /ColorName (MAGENTA) def
37     MAGENTA setupcolor
38     /WhichColor 2 def } if
39   seppages 3 eq {
40     /ColorName (YELLOW) def
41     YELLOW setupcolor
42     /WhichColor 1 def } if
43   seppages 4 eq {
44     /ColorName (black) def
45     black setupcolor
46     /WhichColor 0 def } if

```

Set crop marks on each page with the separation name printed:

```

47 gsave .3 setlinewidth
48 3 -7 moveto
49 /Helvetica findfont 6 scalefont setfont
50 ColorName show
51 0 0 CM
52 vsize cX 2 mul sub dup hsize cX 2 mul sub dup isls{4 2 roll}if 0 CM
53 exch CM 0
54 exch CM
55 grestore 0 cX -2 mul TR isls
56 {cX -2 mul 0 TR}if
57   } def end
58 /separations 48 dict def
59 separations begin
60   /cmykprocs [
61     { pop pop pop SetGray }      % cyan
62     { pop pop exch pop SetGray } % magenta
63     { pop 3 1 roll pop pop SetGray } % yellow
64     { 4 1 roll pop pop pop SetGray } % black
65   ] def
66   /rgbprocs [ %def

```

```

67     { ToCMYK pop pop pop SetGray }
68     { ToCMYK pop pop exch pop SetGray }
69     { ToCMYK pop 3 1 roll pop pop SetGray }
70     { ToCMYK 4 1 roll pop pop pop SetGray }
71   ] def
72   /screenangles [ %def
73     105 % cyan
74     75  % magenta
75     0  % yellow
76     45  % black
77   ] def
78 end

```

The command `setupcolor` takes 0, 1, 2, or 3 as its argument, for cyan, magenta, yellow, and black, respectively:

```

79 /CYAN 0 def           /MAGENTA 1 def
80 /YELLOW 2 def         /black 3 def
81 /setupcolor{ %def
82   userdict begin
83     dup separations /cmypprocs get exch get
84     /setcmkcolor exch def
85     dup separations /rgbprocs get exch get
86     /setrgbcolor exch def
87     separations /screenangles get exch get
88     currentscreen
89     exch pop 3 -1 roll exch
90     setscreen
91     /setscreen { pop pop pop } def

```

We redefine `setgray` so that it shows only on the black separation:

```

92   /setgray {
93     WhichColor 0 eq
94     {systemdict begin adjustdot setgray end}
95     {pop systemdict begin 1 setgray end}
96     ifelse}def
97   end
98 } bind def
99 /adjustdot { dup 0 eq { } { dup 1 exch sub .1 mul add} ifelse } def

```

Finally, we set the percentages of undercolor removal:

```

100 /magentaUCR .3 def
101 /yellowUCR .07 def
102 /blackUCR .4 def

```

For greater control over the separations Graham Freeman's *aurora* package can also be used with *dvips* or any other dvi-to-PostScript program that supports the downloading of header files. This also works by redefining PostScript color commands, but the pages for each different C, M, Y, and K color are produced with separate header files. One main header file (*aurora.pro*) is downloaded together with one of *cyan.pro*, *magenta.pro*, *yellow.pro*, or *black.pro*. Thus the *dvips* command line

Cyan	Warning!	Yellow	Warning!
Magenta		black	Warning!

Figure 22.12: Example of simple color separations.

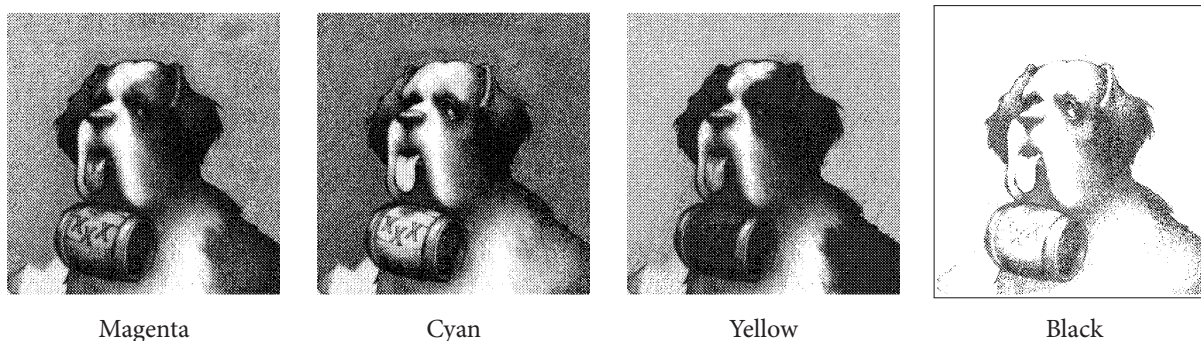


Figure 22.13: Color separation of a bitmap image using aurora.

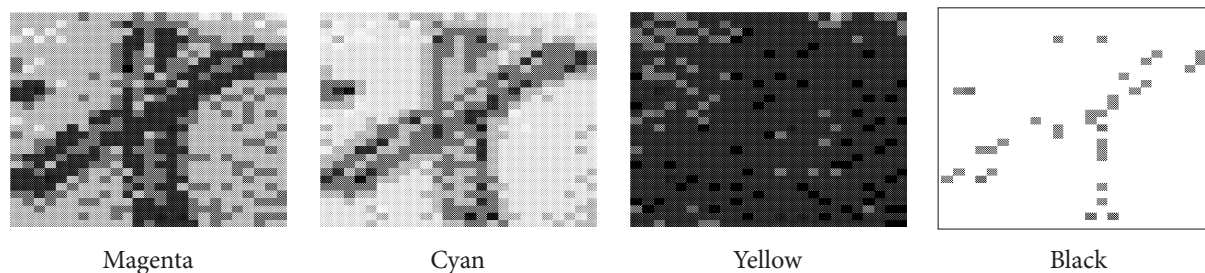


Figure 22.14: Detail of color separation of a bitmap image.

to put the magenta portions of a file `text.dvi` into a PostScript file called `text-magenta.ps` would be

```
dvips text -h aurora.pro -h magenta.pro -o text-magenta.ps
```

Many forms of color bitmap images are handled by *aurora*, but not the full range of possibilities supported by the PostScript Level 2 `colorimage` command; specifically, compound color does not work. Figure 22.13 shows what the separations for an image look like using *aurora*. The familiar canine is shown with a gray shade for the quantities of cyan, magenta, yellow or black. The effect is even clearer in Figure 22.14, a small portion of the same picture (part of the barrel at the dog's neck).

A custom color can be produced by precisely defining the C, M, Y, and K components and providing a matching header file for the color that specifies a point in CMYK space of the separation you want. If the current color in the document fits the color, then the object is printed. This method of selecting custom colors for output requires that the numbers specifying the custom color precisely match the color used within the document. If the color does not match, no output is produced, while with the normal system the component of the current process color determines the *amount* of output to produce.

### 22.7.3 Overprinting

As explained previously, color printing involves generating one image “plate” for each “printing ink”, most often, in four-color mode, cyan, magenta, yellow and black (CMYK). Sometimes one or more spotcolors are added.

When printing black over a colored background, color separation software usually sets the other plates to white. This, however, results in any misregistration on the offset printer generating distracting slivers of white. With a light background this effect can be more or less ignored. If the effect is too visible and it has to be corrected, one can use *overprinting*, i.e., one lets the color continue underneath the black.

Siep Kroonenberg developed the overprint package (available from <http://tex.aanhet.net/overprint/>), which sets and unsets overprinting. It is compatible with both dvips and pdftex. A small example follows. Note that the overprint package loads the color package.

Exa.  
22-7-2



```
\usepackage{overprint}
% All colors explicitly defined as cmyk colors
\definecolor{mg}{cmyk}{0,1,0,0}
\definecolor{cn}{cmyk}{1,0,0,0}
\definecolor{bk}{cmyk}{0,0,0,1}
\color{bk}\Large
\colorbox{cn}{text}
\colorbox{mg}{\overprint text}
\colorbox{cn}{text}
```





# Bibliography

- [1] Adobe Systems. Adobe Type 1 Font Format. Addison-Wesley, Reading, MA, 1990.  
This so-called “White Book” contains the specification of Adobe’s Type 1 font format, including information about hints, the encryption mechanism, encodings, and the flex procedure. Available electronically from  
[http://partners.adobe.com/public/developer/en/font/T1\\_SPEC.PDF](http://partners.adobe.com/public/developer/en/font/T1_SPEC.PDF)
- [2] Adobe Systems Inc. PostScript Language Reference Manual, Third Edition. Addison-Wesley, Reading, MA, 1999.  
This so-called “Red Book” describes the syntax and semantics of the complete PostScript language. The book documents the imaging model and the graphics, fonts, device, and rendering operators. Available electronically from  
<http://www.adobe.com/products/postscript/pdfs/PLRM.pdf>
- [3] Adobe Systems Inc. PDF Reference (Version 1.6), Fifth Edition. Addison-Wesley, Reading, MA, 2005.  
This is the specification of Adobe’s Portable Document Format (PDF). The book introduces and explains all aspects of the PDF format, including its architecture and imaging model (allowing transparency and opacity for text, images, and graphics), the command syntax, the graphics operators, fonts, and rendering, and the relation between PostScript and PDF.  
<http://partners.adobe.com/public/developer/en/pdf/PDFReference16.pdf>
- [4] Adobe Systems Inc. (Glenn C. Reid). PostScript Language Program Design. Addison-Wesley, Reading, MA, 1988.  
This so-called “Green Book” introduces programming techniques for designing efficient PostScript programs with the help of examples in the areas of typesetting text, constructing graphics, writing calculators, debugging programs, etc. These directly usable examples accomplish specific practical tasks and have been carefully designed and debugged to show in detail how the language works. Each of the fifteen chapters addresses a specific aspect of top-to-bottom program design or problem solving and contains some useful advice. Available electronically from  
[http://partners.adobe.com/public/developer/ps/sdk/sample/index\\_psbooks.html](http://partners.adobe.com/public/developer/ps/sdk/sample/index_psbooks.html)
- [5] Angus Duggan. “Colour separation and PostScript”. *TUGboat*, 15(3):213–217, 1994.  
Description of a simple color separator for DVI files, The implementation is based on the color support provided by dvips via the `\special` command, and its limitations are explained. The procedure outputs a DVI file for each of the input colors.  
<http://www.tug.org/TUGboat/Articles/tb15-3/tb44duggan.pdf>
- [6] Michel Goossens, Sebastian Rahtz, Eitan M. Gurai, Ross Moore, and Robert S. Sutor. The  $\text{\LaTeX}$  Web Companion: Integrating  $\text{\LaTeX}$ , HTML, and XML. Addison-Wesley, Reading, MA, 1999.  
This book teaches (scientific) authors how to publish on the Web or other hypertext presentation systems, building on their experience with  $\text{\LaTeX}$  and taking into account their specific needs in fields such as mathematics, non-European languages, and algorithmic graphics. The book explains how to make full use of the Adobe Acrobat format from  $\text{\LaTeX}$ , convert legacy documents to HTML or XML, make use of math in Web applications, use  $\text{\LaTeX}$  as a tool in preparing Web pages, read and write simple XML/SGML, and produce high-quality printed pages from Web-hosted XML or HTML pages using  $\text{\LaTeX}$  or PDF.

- [7] Alan Hoenig. *T<sub>E</sub>X Unbound: Strategies for Fonts, Graphics, and More*. Oxford University Press, New York, 1998.
- This book describes how to produce good typography with  $\text{\TeX}$ , in particular how to set up and make proper use of PostScript fonts, and create high-quality graphics illustrations with  $\text{\TeX}$ -friendly methods. It contains many examples and summaries of procedures to follow.
- The book starts with a good overview of  $\text{\TeX}$ ,  $\text{\LaTeX}$ , MetaFont, and MetaPost, explaining how they all fit together. The second part of the book describes  $\text{\TeX}$ 's font mechanisms. The author does not limit himself to a description of how to set up a standard font family, but includes a lot of more advanced material. Examples included are using special effect fonts, specifying font families that contain alternate character sets or symbols, integrating high-quality commercial fonts, and typesetting mathematics with fonts other than the original  $\text{\TeX}$  fonts (there is a 30-page overview on how to combine available mathematics font families with various often-used typefaces). The final part of the book discusses graphics applications, in particular MetaFont, MetaPost, PSTricks, P<sub>CT</sub> $\text{\TeX}$ , and mpic.
- [8] David Kastrup. *preview-latex*, 2003.
- preview-latex* allows appropriately selected parts of a  $\text{\TeX}$  document to be formatted and displayed within your Emacs editor, allowing you to view what it looks like while still allowing you to edit it. On CTAN at: [support/preview-latex/](http://support/preview-latex/)
- [9] Donald E. Knuth. *Computer Modern Typefaces*, volume E of *Computers and Typesetting*. Addison-Wesley, Reading, MA, 1986.
- This book depicts graphically more than 500 Greek and Roman letterforms, together with punctuation marks, numerals, and many mathematical symbols. The MetaFont code to generate each glyph is given and it is explained how, by changing the parameters in the MetaFont code, all characters in the Computer Modern family of typefaces can be obtained.
- [10] Donald E. Knuth. “Virtual fonts: more fun for grand wizards”. *TUGboat*, 11(1):13–23, 1990.
- The original description of virtual fonts. Initially foreseen as a convenient way to specify a mapping from  $\text{\TeX}$ 's notion of a font character to a device's capabilities for printing, Knuth extended this notion so that arbitrary DVI commands (including rules and even `\specials`) could be part of a virtual font. To help with the creation of virtual fonts two new programs, *vftovp* and *vptovf*, are introduced, as well as the VPL property list which extends the ordinary PL format. Virtual fonts make it easy to go from DVI files to the font layouts of any manufacturer or font supplier.
- <http://www.tug.org/TUGboat/Articles/tb11-1/tb27knut.pdf>
- [11] Donald E. Knuth and Michael F. Plass. “Breaking paragraphs into lines”. *Software, Practice and Experience*, 11:1119–1184, 1981. (Reprinted in Donald E. Knuth, *Digital Typography*, CSLI Lecture Notes Nr. 78, 1999, Ch. 3, pp. 67–155.).
- [12] Gerard Kunkel. *Graphic Design with PostScript*. Scott, Foresman, Glenview, IL, 1990.
- This book is a hands-on guide to using PostScript containing complete coded examples for many practically relevant applications, including (pseudo) 3-D effects for graphs, etc.
- [13] Henry McGilton and Mary Campione. *PostScript by Example*. Addison-Wesley, Reading, MA, 1992.
- This book first introduces the basic concepts of PostScript language (paths, graphic states, text, clipping, transformations, arcs, curves, and images). It then presents a set of tools to construct fonts, patterns, forms, and manage your printing environment. PostScript Level 2 issues such as patterns, forms, images, composite fonts, halftones, and color models are covered. With its many hands-on exercises and step-by-step instructions, this book becomes a genuine toolkit, for building effective PostScript programs.
- [14] Thomas Merz. *PostScript & Acrobat/PDF*. Springer Verlag, Berlin, 1996.
- [15] Frank Mittelbach, Michel Goossens, Johannes Braams, David Carlisle, and Chris Rowley. *The  $\text{\TeX}$  Companion*, Second Edition. Addison-Wesley, Reading, MA, 2004.
- This book describes over 200  $\text{\TeX}$  packages and presents a whole series of tips and tricks for using  $\text{\TeX}$  in both traditional and modern typesetting, in particular how to customize layout features to your own needs—from phrases and paragraphs to headings, lists, and pages. It provides expert advice on using LaTeX's basic formatting tools to create all types of publication, from memos to encyclopedias. It covers in depth important extension packages for tabular and technical typesetting, floats and captions, multi-column layouts, including reference guides and discussion of the underlying typographic concepts. It details techniques for generating and typesetting indexes, glossaries, and bibliographies, with their associated citations.
- [16] Rolf Niepraschk. “Anwendungen des  $\text{\TeX}$ -pakets *preview*”. *Die  $\text{\TeX}$ nische Komi die*, 1/2003:60–65, 2003.
- This article describes how PostScript-related code can be integrated into sources, which will be compiled with pdf $\text{\TeX}$ .

- 
- [17] Michael Sofka. “Color Book Production Using T<sub>E</sub>X”. *TUGboat*, 15(3):228–238, 1994.  
The color separation process in color book production is described, in particular the two commonly used methods: custom color and process color separation. It is shown how T<sub>E</sub>X can be used, together with a suitable DVI driver, to produce high-quality custom and process color books.  
<http://www.tug.org/TUGboat/Articles/tb15-3/tb44sofka.pdf>
- [18] Friedhelm Sowa. “Printing colour pictures”. *TUGboat*, 15(3):223–227, 1994.  
A device independent approach to printing colored T<sub>E</sub>X documents is described. It can be implemented on PostScript, as well as on cheap inkjet color printers.  
<http://www.tug.org/TUGboat/Articles/tb15-3/tb44sowa.pdf>
- [19] Sivan Toledo. “Exploiting rich fonts”. *TUGboat*, 21(2):121–129, 2000.  
Rich fonts containing thousands of glyphs are becoming widely available bundled with operating systems, printers, and software packages. Such fonts can offer better typographic results than conventional fonts. The paper explores three kinds of rich fonts, PostScript Type 3, WGL4 (Windows Glyph List 4) is a glyph list published by Microsoft and intended to support all European languages, including Greek and Russian, and Palatino Linotype, a WGL4 family with an extended set of Latin that allow more typographic options. For each type their typographic features, and ways to exploit these features in T<sub>E</sub>X are described.  
<http://www.tug.org/TUGboat/Articles/tb21-2/tb67tole.pdf>
- [20] The Unicode Consortium. The Unicode Standard, Version 5.0. Addison-Wesley, Reading, MA, 2007.  
The reference guide of the Unicode Standard, a universal character-encoding scheme that defines a consistent way of encoding multilingual text. Unicode is the default encoding of HTML and XML. The book explains the principles of operation and contains images of the glyphs for all characters presently defined in Unicode.  
[Available for restricted use from: http://www.unicode.org/versions/Unicode5.0.0/](http://www.unicode.org/versions/Unicode5.0.0/)



# Index of Commands and Concepts

The index has been split into two parts. We start with a general index that covers all entries. We end with an index of authors.

To make the indexes easier to use, the entries are distinguished by their “type”, and this is often indicated by one of the following “type words” at the beginning of the main entry or a sub-entry:

boolean, counter, document class, env., file, file extension, font, key, key value, option, package, program, rigid length, or syntax.

The absence of an explicit “type word” means that the “type” is either a  $\TeX$  “command” or simply a “concept”.

Use by, or in connection with, a particular package is indicated by adding the package name (in parentheses) to an entry or sub-entry. There is one “virtual” package name, `tlgc`, that indicates commands introduced only for illustrative purposes in this book.

A *blue italic* page number indicates that the command or concept is demonstrated in an example on that page.

When there are several page numbers listed, **bold** face indicates a page containing important information about an entry, such as a definition or basic usage.

When looking for the position of an entry in the index, you need to realize that, when they come at the start of a command or file extension, both the characters `\` and `.` are ignored. All symbols come before all letters and everything that starts with the `@` character will appear immediately before `A`.

## Symbols

\( (pst-pdf), 149  
 \) (pst-pdf), 149  
 + syntax (dvips), 106  
 .dvipsrc file (dvips), 98  
 \/, 23  
 < syntax (dvips), 110, 111  
 « syntax (dvips), 111  
 <[ syntax (dvips), 111  
 = syntax (dvips), 99  
 #copies (PostScript), 102  
 % syntax (dvips), 103  
 %b syntax (dvips), 103  
 %d syntax (dvips), 103  
 %f syntax (dvips), 103  
 %m syntax (dvips), 103  
 %p syntax (dvips), 103  
 ` syntax (dvips), 103  
 8r.enc file, 26, 34, 35, 110

## @

@ syntax (dvips), 106

## A

abx option (MnSymbol), 64  
 Acrobat program, 103, 163, 166  
 Acrobat 3 program, 131  
 Acrobat 4 program, 131  
 Acrobat 5 program, 131  
 Acrobat Distiller program, 131, 132, 147  
 Acrobat Reader program, 90, 91, 153  
 active option (pst-pdf), 149  
 Adobe FrameMaker program, 169  
 Adobe Illustrator program, 117, 182  
 Adobe Photoshop program, 182  
 Adobe Type Manager program, 97, 122  
 .afm file extension, 18, 19–23  
 All Commands  
   CHARACTER (PL/VPL), 24  
   FONTAT (PL/VPL), 25  
   FONTCHECKSUM (PL/VPL), 24  
   FONTDSIZE (PL/VPL), 25  
   FONTNAME (PL/VPL), 24  
   MAP (PL/VPL), 24  
   MAPFONT (PL/VPL), 24  
   VTITLE (PL/VPL), 24  
 amsbb option (MinionPro), 65  
 amssymb package, 64  
 amsmath package, 63  
 amssymb package, 64  
 \AtBeginDvi, 106  
 aurora program, 187, 188  
 aurora.pro file, 187  
 autodistinglist env. (pifont), 40  
 autofont program, 69  
 avant package, 36, 38

## B

babel package, 66  
 bibtex program, 150, 154  
 bind (PostScript), 137

black.pro file, 187  
 bookman package, 36, 37  
 bop-hook (PostScript), 102, 111, 114, 115

## C

center (PostScript), 115  
 \centering, 42  
 cfftot1 program, 45, 59  
 chancery package, 36  
 charter package, 36, 43  
 CJK package, 52, 53  
 cmsy option (MnSymbol), 64  
 \color, 189  
 color package, 108, 184, 189  
 \colorbox, 189  
 colorimage (PostScript), 188  
 colorsep.pro file (dvips), 185  
 config.lino file (dvips), 107  
 config.ps file (dvips), 98, 101  
 convert program, 155  
 Corel Draw program, 182  
 courier package, 36, 37  
 cvs (PostScript), 115  
 cyan.pro file, 187

## D

\DeclareFontFamily, 34  
 \DeclareFontshape, 34  
 \definecolor, 189  
 deletefile (PostScript), 121  
 diff program, 181  
 \ding (pifont), 38, 40  
 dingautolist env. (pifont), 38  
 \dingfill (pifont), 40  
 \dingline (pifont), 40  
 dinglist env. (pifont), 38  
 displaymath env. (pst-pdf), 149  
 displaymath option (pst-pdf), 149  
 \dotfill, 40  
 draft option (pst-pdf), 149  
 draftcopy package, 112  
 draftwatermark package, 112  
 dup (PostScript), 115  
 .dvi file extension, 100, 102, 105, 106, 110, 111  
 dvi2svg program, 91, 97  
 dvipdf program, 133  
 dvipdfm program, 147, 148, 152  
 dvipdfm-jpn program, 147  
 dvipdfm-kor program, 147  
 dvipdfmx program, 147, 148, 151–155  
 dvips program, x, 1, 25, 26, 32, 34, 41, 43, 56, 73, 87, 97, 98–116, 133, 144, 146–155, 184–187, 189  
 dvips.ini file (dvips), 98  
 dvipsone program, 97  
 dvisvg program, 91  
 dvisvgm program, 91  
 dviwindo program, 97

## E

emacs program, 161  
 \emph, 23  
 .enc file extension, 110

Encoding (PostScript), 31  
 end-hook (PostScript), 111, 135  
 endmulti (PostScript), 136  
 Environment Variables  
   DVIPSHEADERS (dvips), 103  
   GS\_DEVICE (Ghostscript), 123  
   GS\_FONTPATH (Ghostscript), 120, 122  
   GS\_LIB (Ghostscript), 121  
   PRINTER (dvips), 98, 100–102  
 eop-hook (PostScript), 111  
 epsffit program, 138, 144  
 .epsi file extension, 130  
 epstool program, 100, 133  
 epstopdf program, 93, 134, 152, 155  
 eqnarray env. (pst-pdf), 149  
 equation env. (pst-pdf), 149  
 everypage package, 112, 175  
 Evince program, 91, 129  
 evince program, 129, 130, 176  
 exch (PostScript), 115  
 ExtendFont (PostScript), 26  
 ExtendFont (PostScript), 110  
 extractres program, 138, 143, 144

## F

\familydefault, 38  
 fancyhdr package, 175  
 \fboxrule rigid length, 92  
 \fboxsep rigid length, 92  
 \fcolorbox, 92  
 .fd file extension, 34, 35  
 \figureversion, 64  
 final option (pst-pdf), 149  
 firstpage option (draftwatermark), 112  
 fixfmeps program, 138  
 fixwfmeps program, 138  
 fixwmps program, 138  
 fixwmps program, 138  
 Flash program, 134  
 \FlipPDF (pst-pdf), 175  
 fntguide.tex file, 36  
 \fontdimen, 23  
 Fontmap file (Ghostscript), 122  
 \fontsize, 42  
 FontTools program, 16  
 footnotefigures option (MinionPro), 65  
 fourier package, 36, 41  
 fourierbb option (MinionPro), 65  
 FrameMaker program, 130, 134, 138, 173  
 frenchmath option (MinionPro), 65  
 FullName (PostScript), 31

## G

\geometry (pst-pdf), 171–173  
 geometry package, 171–173, 175  
 getafm program, 138  
 .gf file extension, 30  
 gftodvi program, 30  
 gftopk program, 30  
 gftype program, 30  
 ggv program, 129  
 Ghostscript program, ix, 88, 116–201

ghostscript program, x, 36, 87, 91, 129, 131, 133, 134  
 GhostView program, ix  
 ghostview program, x, 89, 129, 130, 132, 133, 153, 156, 176  
 Gnome program, 129  
 gnuplot program, 134, 146  
 graphicx package, 108, 146, 149, 170, 171  
 grep program, 139  
 groff program, 134  
 gs program, 125  
 gs\_init.ps file, 125  
 gsftopk program, 101  
 GSview program, 129  
 gv program, 129

## H

helvet package, 36, 37, 38  
 hhline package, 101  
 hmirror (PostScript), 113, 114  
 hsize (PostScript), 113  
 hyperref package, 147, 152, 153, 167  
 hypertext package, 153

## I

ldraw program, 134  
 idraw program, 146  
 ifthen package, 175  
 Illustrator program, 131  
 ImageMagick program, 126, 134  
 inactive option (pst-pdf), 149  
 \includegraphics  
   (graphics), 146  
   (pst-pdf), 170  
 \includepdf (pst-pdf), 170, 171–173  
 includeres program, 138, 143, 144  
 italicgreek option (MinionPro), 65  
 \itGamma, 65  
 \itgamma, 65

## J

java program, 165, 167, 177  
 .jpeg file extension (pst-pdf), 154

## L

\label (pst-pdf), 171  
 latex program, 147, 149, 151–155, 173  
 lf option (MinionPro), 64  
 lpr program, 125  
 lucidabb option (MinionPro), 65  
 LY1 font encoding, 56, 57

## M

Mac GS Viewer program, 129  
 magenta.pro file, 187  
 makeindex program, 154  
 MakeTeXPK program, 98  
 .map file extension, 26, 34  
 \mathbb, 65  
 \mathcal, 65  
 Mathematica program, 134  
 mathlf option (MinionPro), 64  
 mathosf option (MinionPro), 64

mathpazo package, 36, 38  
 mathpple package, 36  
 mathptm package, 36  
 mathptmx package, 36, 37, 38  
 mathtabular option (MinionPro), 64  
 MATLAB program, 146  
 mbtPDFasm program, 162, 178  
 mbtPdfAsm program, 162–165  
 MetaPost program, 134  
 Microsoft Office program, 169  
 Microsoft Word program, 138  
 minionint option (MinionPro), 65  
 MinionPro package, 64, 66, 67, 74  
 minionpro package, 59  
 missfont.log file, 101  
 mixedgreek option (MinionPro), 65  
 mnsy option (MnSymbol), 64  
 MnSymbol file, 63  
 MnSymbol package, 63–65  
 multi (PostScript), 136  
 multi.pro file, 135  
 Multivalent program, 165, 166  
 multivalent program, 177, 179  
 myfont.ttx file, 17  
 myfontmods.ttx file, 17  
 mystring (PostScript), 115

## N

\ncdiag, pnode (pst-pdf), 150  
 netpbm program, 126  
 newcent package, 36  
 nopstricks option (pst-pdf), 149  
 notightpage option (pst-pdf), 149

## O

off option (pdfflip), 175  
 Office program, 12  
 Omega program, 102  
 Open Office program, 169  
 OpenOffice program, 13  
 Openoffice program, 12  
 osf option (MinionPro), 164  
 ot1ptm.fd file, 35  
 otfinfo program, 14, 55, 58  
 otfmtotfm program, 56  
 otftotfm program, 53, 55, 57, 58  
 \overprint, 189  
 overprint package, 189  
 OzTeX program, 97

## P

### packages

P<sub>CT</sub>TeX, 198  
 amsfonts, 64  
 amsmath, 63  
 amssymb, 64  
 avant, 36, 38  
 babel, 66  
 bookman, 36, 37  
 chancery, 36  
 charter, 36, 43  
 CJK, 52, 53

### packages (cont.)

color, 108, 184, 189  
 courier, 36, 37  
 draftcopy, 112  
 draftwatermark, 112  
 everypage, 112, 175  
 fancyhdr, 175  
 fourier, 36, 41  
 geometry, 171–173, 175  
 graphicx, 108, 146, 149, 170, 171  
 helvet, 36, 37, 38  
 hline, 101  
 hyperref, 147, 152, 153, 167  
 hypertext, 153  
 ifthen, 175  
 mathpazo, 36, 38  
 mathpple, 36  
 mathptm, 36  
 mathptmx, 36, 37, 38  
 MinionPro, 64, 66, 67, 74  
 minionpro, 59  
 MnSymbol, 63–65  
 newcent, 36  
 overprint, 189  
 palatino, 36  
 pdfflip, 175  
 pdfpages, 169–173, 175  
 pifont, 38–41, 66  
 preview, 149–151  
 psfrag, 145, 146, 147, 151  
 PSNFSS, 26, 32–35, 67  
 pst-pdf, 147, 149–151, 153, 154  
 pstricks, 147, 149  
 textcomp, 63  
 times, 32, 36, 37, 38  
 tlgc, 201  
 utopia, 36, 41

### \pagestyle, 92

palatino package, 36  
 PCT<sub>CT</sub>TeX program, 97  
 .pdf file extension (pst-pdf), 154  
 pdf2ps program, 131, 132  
 pdf90 program, 173  
 pdfcrop program, 135, 152  
 pdfflip package, 175  
 pdffonts program, 179, 180  
 pdfimages program, 181  
 pdfinfo program, 152, 165, 178, 179  
 pdfjoin program, 173  
 pdflatex program, ix, x, 2, 43, 46, 47, 49, 51, 53, 112, 147, 149–155, 169, 173, 180, 181  
 pdftex program, 173  
 pdfmark (PostScript), 103, 121  
 pdfnup program, 173  
 pdfopt program, 132  
 pdfpages package, 169–173, 175  
 pdftex program, 180, 189  
 pdftex.map file, 43  
 Pdftk program, 154  
 pdftk program, 93, 155, 157–159, 162, 163  
 pdftopbm program, 181  
 pdftoppm program, 181  
 pdftops program, 155, 177



pdftosrc program, 180, 181  
 pdftotext program, 177  
 .pfa file extension, 31, 32, 110  
 .pfb file extension, 31, 32, 44, 110  
 Piautolist env. (pifont), [40](#)  
 P<sub>CT</sub>E<sub>X</sub> package, 198  
 picture env. (psfrag), 146  
 \Piifill, [66](#)  
     (pifont), 40, [41](#)  
 pifont package, 38–41, 66  
 \Piline, [66](#)  
     (pifont), 40, [41](#)  
 Pilist env. (pifont), 40  
 \Pisymbol (pifont), 40  
 .pk file extension, 30, 32, 34, 100, 101, 103, 109, 135  
 pktogf program, 30  
 pktype program, 30  
 .pl file extension, 23, 24  
 pltotf program, 23  
 .png file extension (pst-pdf), 154  
 postscript env. (pst-pdf), 151  
 preview package, 149–151  
 \PreviewEnvironment (pst-pdf), 150  
 print program, 100  
 .ps file extension, 100  
 ps2ai.ps file (Ghostscript), 131  
 ps2ascii program, 129  
 ps2ascii.ps file (Ghostscript), 129, 130  
 ps2epsi program, 130, 131  
 ps2pdf program, 97, 103, 131, 132, 147, 150, 151, 153–155  
 ps2pdf12 program, 131  
 ps2pdf13 program, 131, 133, 153  
 ps2pdf14 program, 131  
 ps2pdfwr program, 131  
 ps2pk program, 101  
 ps2ps program, 132  
 psbook program, 137, 138, 141, 142  
 psfonts.map file, 43  
     (dvips), 102, 109, 110, 111  
 \psfrag (psfrag), 145, 146  
 psfrag package, 145, 146, 147, 151  
 psmatrix env. (pst-pdf), 149  
 psmerge program, 138, 142  
 PSNFSS package, 26, 32–35, 67  
 psnup program, 137, 138, 139, 140, 172  
 pspicture env. (pst-pdf), 149  
 psresize program, 138, 143  
 psselect program, 138, 142  
 pst-pdf package, 147, 149–151, 153, 154  
 \pst@object (pst-pdf), 149  
 \pstextpath (pst-pdf), [150](#)  
 pstoeit program, 134  
 pstops program, 137–140, 172  
 pstotext program, 130  
 pstricks option (pst-pdf), 149  
 pstricks package, 147, 149  
 psutils program, 135–145, 166

## R

ReEncodeFont (PostScript), 26, 110  
 renamefile (PostScript), 121  
 Resolution (PostScript), 109

\rmdefault, 32, 34, [37](#), [92](#)

## S

scale (PostScript), 113  
 scaled option (helvet), 37  
 SDict (PostScript), 109  
 Separator program, 182  
 setpagedevice (PostScript), 114  
 setrgbcolor (PostScript), 162  
 \sfdefault, [37](#), [38](#)  
 show (PostScript), 115  
 showchar program, 138  
 showpage (PostScript), 118, 135, 145  
 SlantFont (PostScript), 26  
 Slantfont (PostScript), 110  
 \slantfrac, [66](#)  
 \smallfrac, [66](#)  
 \special, 23, 103, 105, 108, 109  
     (psfrag), 146, 147  
 StandardEncoding (PostScript), 31  
 start-hook (PostScript), 111, 135  
 stdin file (Ghostscript), 119  
 systemdict (PostScript), 118, 119

## T

T1 font encoding, 25, 26, 56  
 t1ascii program, 31  
 t1asm program, 31  
 t1binary program, 31  
 t1disasm program, 31  
 t1dotlessj program, 60  
 t1mac program, 32  
 t1ptm.fd file, 34, 35  
 t1testpage program, 60  
 t1unmac program, 32  
 t1utils program, 31  
 T2 font encoding, 26  
 T7 font encoding, 26  
 tabular env. (pst-pdf), [150](#)  
 teTeX program, 45  
 TeXBase1Encoding (PostScript), 26, 27, 28, 34, 110  
 texhash program, 57  
 texnansx font encoding, 56  
 \textbf, 34  
 textcomp package, 63  
 \textheight rigid length, 106  
 \textit, 23  
 textlf option (MinionPro), 64  
 textosf option (MinionPro), 64  
 \textsc, [65](#)  
 \textsf, 37  
 \textssc, [65](#)  
 \textsw, [65](#)  
 Textures program, 97, 184  
 \textwidth rigid length, 106  
 .tfm file extension, 18, 23, 24, 30, 35, 100, 103, 110  
 tftopl program, 23  
 tgif program, 134  
 tightpage option (pst-pdf), 149  
 times package, 32, 36, 37, 38  
 tlgc package, 201  
 tpic program, 98, 108

translate (PostScript), 113

TrueTeX program, 46

.ttc file extension, 18

ttc2ttf program, 18

\ttdefault, 37

ttf2tfm program, 49

ttx program, 16, 17

## U

\UnFlipPDF (pst-pdf), 175

updmap program, 57

\upGamma, 65

\upgamma, 65

\uput (pstricks), 137

\usefont, 41, 42

userdict (PostScript), 101, 111, 118

utopia package, 36, 41

## V

.vf file extension, 24, 30, 35, 103

vftovp program, 24

vmirror (PostScript), 113, 114

.vpl file extension, 24

vptovf program, 24, 49

vsize (PostScript), 113

VTeX program, 97, 147

## W

Windows Write program, 138

Word program, 173

WordPerfect program, 138

## X

xfig program, 134, 146

xpdf program, 91, 152, 176, 180

## Y

yellow.pro file, 187

## Z

zlib program, 148

# People

Bühmann, Andreas, 59  
Barratt, Craig, 145  
Berry, Karl, 27, 37, 67  
Birrell, Andrew, 130  
Blumensath, Achim, 59, 63

Caignaert, Christophe, 46, 47, 49  
Callegari, Sergio, 112, 175  
Carlisle, David, 145  
Cartlidge, Ross, 135  
Cho, Jin-Hwan, 147

Deutsch, L. Peter, 116  
Duggan, Angus, 137

Firth, David, 173  
Freeman, Graham, 187  
Frischauf, Adrian, 91  
Frutiger, Adrian, 46

Geschke, Charles, 88  
Giesecking, Martin, 91  
Glunz, Wolfgang, 134  
Goossens, Michel, 198, 199  
Grant, Michael, 145

Hàn, Thế Thành, 147, 180

Harders, Harald, 46  
Hartke, Stephen G., 38  
Hetherington, Lee, 31  
Hirata, Shunsaku, 147  
Horn, Berthold, 27

Jackowski, Bogusław, 115  
Jeffrey, Alan, 27

Kinch, Richard J., 46  
Knuth, Donald, 1, 198  
Kohler, Eddie, 31, 55, 58, 69  
Kroonenberg, Siep, 189

Lang, Russell, 129, 133  
Lehman, Philipp, 36  
Lemberg, Werner, 52

MacKay, Pierre, 27  
Matthias, Andreas, 169, 175  
McJones, Paul, 130  
Merz, Thomas, 117  
Mittelbach, Frank, 199  
Mocnik, Jaka, 129

Owens, John, 57, 58

Pepping, Simon, 18  
Plass, Johannes, 129

Rahtz, Sebastian, 27, 36, 38, 198  
Rakityansky, Damir, 49  
Rokicki, Tom, 34, 97, 98

Sabo, Rudolf, 91  
Saunders, Michael, 55  
Schmidt, Walter, 36, 44  
Schmitt, Thierry, 162

Theisen, Timothy, 129  
Toledo, Sivan, 54  
Trevorrow, Andrew, 97

Vollmer, Jürgen, 112  
Vulis, Michael, 97

Warnock, John, 88  
Wicks, Mark A., 147  
Wood, Alan, 5  
Wyart, Damien, ii

Zedler, Michael, 59