Dvips: A DVI-to-PostScript Translator

for version 5.66a
February 1997

Tomas Rokicki
(edited for Dvipsk by `kb@mail.tug.org`)

1 Why use Dvips?

The Dvips program has a number of features that set it apart from other PostScript drivers for TeX. This rather long section describes the advantages of using Dvips, and may be skipped if you are just interested in learning how to use the program. See Chapter 2 [Installation], page 2, for details of compilation and installation.

The Dvips driver generates excellent, standard PostScript, that can be included in other documents as figures or printed through a variety of spoolers. The generated PostScript requires very little printer memory, so very complex documents with a lot of fonts can easily be printed even on PostScript printers without much memory, such as the original Apple LaserWriter. The PostScript output is also compact, requiring less disk space to store and making it feasible as a transfer format.

Even those documents that are too complex to print in their entirety on a particular printer can be printed, since Dvips will automatically split such documents into pieces, reclaiming the printer memory between each piece.

The Dvips program supports graphics in a natural way, allowing PostScript graphics to be included and automatically scaled and positioned in a variety of ways.

Printers with any resolution are supported, even if they have different resolutions in the horizontal and vertical directions. High resolution output is supported for typesetters, including an option that compresses the bitmap fonts so that typesetter virtual memory is not exhausted. This option also significantly reduces the size of the PostScript file and decoding in the printer is very fast.

Missing fonts can be automatically generated if Metafont exists on the system, or fonts can be converted from GF to PK format on demand. If a font cannot be generated, a scaled version of the same font at a different size can be used instead, although Dvips will complain loudly about the poor aesthetics of the resulting output.

Users will appreciate features such as collated copies and support for '`tpic`', '`psfig`', '`emtex`', and '`METAPOST`'; system administrators will love the support for multiple printers, each with their own configuration file, and the ability to pipe the output directly to a program such as '`lpr`'. Support for MS-DOS, OS/2, and VMS in addition to Unix is provided in the standard distribution, and porting to other systems is easy.

One of the most important features is the support of virtual fonts, which add an entirely new level of flexibility to TeX. Virtual fonts are used to give Dvips its excellent PostScript font support, handling all the font remapping in a natural, portable, elegant, and extensible way. Dvips even comes with its own Afm2tfm program that creates the necessary virtual fonts and TeX font metric files automatically from the Adobe font metric files.

Source is provided and freely distributable, so adding a site-specific feature is possible. Adding such features is made easier by the highly modular structure of the program.

There is really no reason to use another driver, and the more people use Dvips, the less time will be spent fighting with PostScript and the more time will be available to create beautiful documents. So if you don't use Dvips on your system, get it today.

Tom Rokicki wrote and maintains the original Dvips program.

2 Installation

(A copy of this chapter is in the distribution file '`dvipsk/INSTALL`'.)

Installing Dvips is mostly the same as installing any Kpathsea-using program. Therefore, for the basic steps involved, see section "Installation" in *Kpathsea*. (A copy is in the file '`kpathsea/INSTALL`'.)

For solutions to common installation problems and information on how to report a bug, see the file '`kpathsea/BUGS`' (see section "Bugs" in *Kpathsea*). For solutions to Dvips-specific problems, see Section 2.4.1 [Debug options], page 5. Also see the Dvips home page at `http://www.radicaleye.com/dvips`.

Dvips does require some additional installation, detailed in the sections below. Also, to configure color devices, see Section 7.5 [Color device configuration], page 49.

2.1 '`config.ps`' installation

Dvips has its own configuration files: a file '`config.ps`' for sitewide defaults, and a file '`config.`*printer*' for each printer (output device). Since these are site-specific, `make install` does not create them; you must create them yourself.

(These Dvips configuration files are independent of the Kpathsea onfiguration file '`texmf.cnf`' (see section "Config files" in *Kpathsea*).

Dvips configuration files contents and searching are described fully in Section 3.4 [Config files], page 15. The simplest way to create a new configuration file is to copy and modify the file '`dvipsk/contrib/config.proto`', seasoning with options to your taste from Section 3.4 [Config files], page 15. Here is '`config.proto`' for your reading pleasure:

```
% Prototype Dvips configuration file.

% How to print, maybe with lp instead lpr, etc.
o |lpr

% Default resolution of this device, in dots per inch.
D 600

% Metafont mode.  (This is completely different from the -M command-line
% option, which controls whether MakeTeXPK is invoked.)  Get
% ftp://ftp.tug.org/tex/modes.mf for a list of mode names.  This mode
% and the D number above must agree, or MakeTeXPK will get confused.
M ljfour

% Memory available. Download the three-line PostScript file:
%   %! Hey, we're PostScript
%   /Times-Roman findfont 30 scalefont setfont 144 432 moveto
%   vmstatus exch sub 40 string cvs show pop showpage
% to determine this number. (It will be the only thing printed.)
m 3500000

% Correct printer offset. You can use testpage.tex from the LaTeX
```

```
% distribution to find these numbers. Print testpage.dvi more than once.
O 0pt,0pt

% Partially download Type 1 fonts by default.  Only reason not to do
% this is if you encounter bugs.  (Please report them to
% tex-k@mail.tug.org if you do.)
j

% Also look for fonts at these resolutions.
R 300 600

% With a high resolution and a RISC cpu, better to compress the bitmaps.
Z

% Uncomment these if you have and want to use PostScript versions of the
% fonts.
%p +cmfonts.map
%p +lafonts.map
%p +cyrfonts.map
%p +eufonts.map

% You will also want definitions for alternative paper sizes -- A4,
% legal, and such.  Examples in ‘contrib/papersize.level2’ and
% ‘contrib/papersize.simple’.
```

2.2 PostScript font installation

To use PostScript fonts with TeX and Dvips, you need both metric files (‘.tfm’ and ‘.vf’) and the outlines (‘.pfa’ or ‘.pfb’). See Section 6.1 [Font concepts], page 34.

To support the basic PostScript font set, the recommended (and simplest) approach is to retrieve `ftp://ftp.tug.org/tex/psfonts.tar.gz` and unpack it in your `$(fontdir)` directory (‘/usr/local/share/texmf/fonts’ by default). This archive contains metrics, outlines, and bitmaps (for previewing) for the 35 de facto standard fonts donated by URW and the additional high-quality freely available PostScript fonts donated by Adobe, Bitstream, and URW, including geometrically-created variants such as oblique and small caps.

‘*CTAN:*/fonts/psfonts’ contains support for many additional fonts for which you must buy outlines (Adobe, Bigelow & Holmes, Monotype, Softkey, Y&Y). ‘psfonts.tar.gz’ is a small extract from this directory. (For CTAN info, see section “unixtex.ftp” in *Kpathsea*; a copy is in the top-level file ‘INSTALL’.)

If you have additional PostScript fonts, you can make them available to Dvips by (1) giving them with appropriate filenames; and (2) running Afm2tfm (see Section 6.2 [Making a font available], page 38) to make TFM and VF metrics for TeX and Dvips to use. Also add them to ‘psfonts.map’ if necessary (see Section 6.4 [psfonts.map], page 45); it contains everything contained in ‘psfonts.tar.gz’ and most fonts that come with Unix systems.

Following are locations for vendor-supplied fonts. Please mail `tex-k@mail.tug.org` if you find fonts elsewhere on your system.

DEC Ultrix
: `/usr/lib/DPS/outline/decwin`

DEC Digital Unix
: `/usr/lib/X11/fonts/Type1Adobe`

HP HP-UX 9, 10
: `/usr/lib/X11/fonts/type1.st/typefaces`

IBM AIX
: `/usr/lpp/DPS/fonts/outlines`
`/usr/lpp/X11/lib/X11/fonts/Type1`
`/usr/lpp/X11/lib/X11/fonts/Type1/DPS`

NeXT
: `/NextLibrary/Fonts/outline`

SGI IRIX
: `/usr/lib/DPS/outline/base /usr/lib/X11/fonts/Type1`

Sun SunOS 4.x
: (NeWSprint only)
`newsprint_2.5/SUNWsteNP/reloc/$BASEDIR/`
`NeWSprint/small_openwin/lib/fonts`
`/usr/openwin/lib/X11/fonts/Type1/outline`

Sun Solaris 2
: `/usr/openwin/lib/X11/fonts/Type1/outline`

VMS
: `SYS$COMMON:[SYSFONT.XDPS.OUTLINE]`

The NeXT system supplies more fonts than any others, but there's a lot of overlap.

Finally, if you have an Hewlett-Packard printer, you should be able to get Type 1 font files for the standard 35 fonts from HP, if the freely available URW Type 1's do not satisfy for whatever reason. The phone number for HP Printer Drivers is (in the United States) 303-339-7009. The driver set to ask for is Adobe Type Manager 2.51, and the disk set number is '`MP210en3`'. Mentioning anything other than Microsoft Windows when you ask for the driver set will likely lead to great confusion on the other end.

2.3 Ghostscript installation

Ghostscript is a PostScript interpreter freely available to end-users, written by Peter Deutsch. It can read the PostScript produced by Dvips and render it on your monitor, or for another device (e.g., an Epson printer) that does not support PostScript, or in PDF format. The latest version is available via `http://www.cs.wisc.edu/~ghost/index.html` and `ftp://ftp.cs.wisc.edu/pub/ghost/aladdin/`.

A somewhat older version of Ghostscript is available under the GNU General Public License, free to everyone. You can get that from `ftp://prep.ai.mit.edu/pub/gnu/`.

The program Ghostview, written by Tim Theisen, provides typical previewing capabilities (next page/previous page, magnification, etc.). It requires Ghostscript to run, and files in structured Postscript, specifically with '`%%Page`' comments (no '`N`' in '`config.ps`'). You can get Ghostview from the same places as Ghostscript.

2.4 Diagnosing problems

You've gone through all the trouble of installing Dvips, carefully read all the instructions in this manual, and still can't get something to work. The following sections provide some helpful hints if you find yourself in such a situation.

For details on effective bug reporting, common installation problems, and `mktexpk` problems, see section "Bugs" in *Kpathsea*.

2.4.1 Debug options

The '`-d`' flag to Dvips helps in tracking down certain errors. The parameter to this flag is an integer that tells what errors are currently being tracked. To track a certain class of debug messages, simply provide the appropriate number given below; if you wish to track multiple classes, sum the numbers of the classes you wish to track. To track all classes, you can use `-1`. Another useful value is `3650`, which tracks everything having to do with file searching and opening.

Some of these debugging options are actually provided by Kpathsea (see section "Debugging" in *Kpathsea*).

The classes are:

1	specials
2	paths
4	fonts
8	pages
16	headers
32	font compression
64	files
128	config files
256	Partial Type 1 font encoding vectors
512	Partial Type 1 subr calls
1024	Kpathsea `stat` calls
2048	Kpathsea hash table lookups
4096	Kpathsea path element expansion
8192	Kpathsea path searches

2.4.2 No output at all

If you are not getting any output at all, even from the simplest one-character file (for instance, '`\ \bye`'), then something is very wrong. Practically any file sent to a PostScript laser printer should generate some output, at the very least a page detailing what error occurred, if any. Talk to your system administrator about downloading a PostScript error handler. (Adobe distributes a good one called '`ehandler.ps`'.)

It is possible, especially if you are using non-Adobe PostScript, that your PostScript interpreter is broken. Even then it should generate an error message. Dvips tries to work around as many bugs as possible in common non-Adobe PostScript interpreters, but doubtless it misses a few. PowerPage Revision 1, Interpreter Version 20001.001, on a Mitsubishi Shinko CHC-S446i color thermal dye sublimation printer is known to be unable to print with any but builtin fonts.

If Dvips gives any strange error messages, or compilation on your machine generated a lot of warnings, perhaps the Dvips program itself is broken. Try using the debug options to determine where the error occurred (see Section 2.4.1 [Debug options], page 5).

It is possible your spooler is broken and is misinterpreting the structured comments. Try the '`-N`' flag to turn off structured comments and see what happens.

2.4.3 Output too small or inverted

If some documents come out inverted or too small, probably your spooler is not supplying an end of job indicator at the end of each file. (This commonly happens on small machines that don't have spoolers.) You can force Dvips to do this with the '`-F`' flag (or '`F`' config file option), but this generates files with a terminating binary character (control-D). You can also try using the '`-s`' flag (or '`s`' config file option) to enclose the entire job in a save/restore pair. See Section 3.2 [Command-line options], page 8, and Section 3.4 [Config files], page 15.

2.4.4 Error messages from printer

If your printer returns error messages, the error message gives very good information on what might be going wrong. One of the most common error messages is '`bop undefined`'. This is caused by old versions of Transcript and other spoolers that do not properly parse the setup section of the PostScript. To fix this, turn off structured comments with the '`-N`' option, but it'd be best to get your spooling software updated.

Another error message is '`VM exhausted`'. Some printers indicate this error by locking up, others quietly reset. This is caused by Dvips thinking that the printer has more memory than it actually does, and then printing a complicated document. To fix this, try lowering the '`m`' parameter in the configuration file; use the debug option to make sure you adjust the correct file.

Other errors may indicate you are trying to include graphics that don't nest properly in other PostScript documents, among other things. Try the PostScript file on a QMS PS-810 or other Adobe PostScript printer if you have one, or Ghostscript (see Section 2.3 [Ghostscript installation], page 4); it might be a problem with the printer itself.

2.4.5 Long documents fail to print

This is usually caused by incorrectly specifying the amount of memory the printer has in the configuration file; see the previous section.

2.4.6 Including graphics fails

The most common problem with including graphics is an incorrect bounding box (see Section 5.1.1 [Bounding box], page 23). Complain to whoever wrote the software that generated the file if the bounding box is indeed incorrect.

Another possible problem is that the figure you are trying to include does not nest properly; there are certain rules PostScript applications must follow when generating files to be included. The Dvips program includes work-arounds for such errors in Adobe Illustrator and other programs, but there are certainly applications that haven't been tested.

One possible thing to try is the '`-K`' flag which strips the comments from an included figure. This might be necessary if the PostScript spooling software does not read the structured comments correctly. Use of this flag will break graphics from some applications, though, since some applications read the PostScript file from the input stream, looking for a particular comment.

Any application which generates graphics output containing raw binary (not ASCII hex) will probably fail with Dvips.

3 Invoking Dvips

Dvips reads a DVI file as output by (for example) TEX, and converts it to PostScript, taking care of builtin or downloaded PostScript fonts, font reencoding, color, etc. These features are described in other chapters in this document.

There many ways to control Dvips' behavior: configuration files, environment variables, and command-line options.

3.1 Basic usage of Dvips

To use Dvips at its simplest, simply type

```
dvips foo
```

where '`foo.dvi`' is the output of TEX that you want to print. If Dvips has been installed correctly, the document will probably roll out of your default printer.

If you use fonts that have not been used on your system before, they may be automatically generated; this process can take a few minutes, so progress reports appear by default. The next time that document is printed, these fonts will have been saved in the proper directories, so printing will go much faster. (If Dvips tries to endlessly generate the same fonts over and over again, it hasn't been installed properly. See section "Unable to generate fonts" in *Kpathsea*.)

Many options are available (see the next section). For a brief summary of available options, just type

```
dvips --help
```

3.2 Command-line options

Dvips has a plethora of command line options. Reading through this section will give a good idea of the capabilities of the driver.

3.2.1 Option summary

Here is a handy summary of the options; it is printed out when you run Dvips with no arguments or with the standard '`--help`' option.

```
Usage: dvips [OPTION]... FILENAME[.dvi]
  Translate the given DVI file to PostScript.

a*  Conserve memory, not time      A   Print only odd (TeX) pages
b # Page copies, e.g., for posters B   Print only even (TeX) pages
c # Uncollated copies              C # Collated copies
d # Debugging                      D # Resolution
e # Maxdrift value                 E*  Create minimal EPSF
f*  Run as filter                  F*  Send control-D at end
h f Add header file f              H f Same as h
i*  Separate file per section
j*  Partially download Type 1's
k*  Print crop marks               K*  Pull comments from inclusions
```

```
l # Last page
m*  Manual feed                       M*  Don't make fonts
n # Maximum number of pages           N*  No structured comments
o f Output file                       O c Set/change paper offset
p # First page                        P s Load config.$s
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*  Disable string param trick
                                      V*  Send downloadable PS fonts as PK
x # Override dvi magnification        X # Horizontal resolution
y # Multiply by dvi magnification  Y # Vertical resolution
z*  Hyperdvi to HyperPostScript       Z*  Compress bitmap fonts
-   Query interactively for options
pp #-# First-last page
mode s Set mode to s

    # = number   f = file   s = string   * = suffix, 0 to turn off
    c = comma-separated dimension pair (e.g., 3.1in,-41.5cm)

Email bug reports to tex-k@mail.tug.org.
```

3.2.2 Option details

Many of the parameterless options listed here can be turned off by suffixing the option with a zero (‘0’); for instance, to turn off page reversal, use ‘-r0’. Such options are marked with a trailing ‘*’.

‘-’ Read additional options from standard input after processing the command line.

‘--help’ Print a usage message and exit.

‘--version’
Print the version number and exit.

‘-a*’ Conserve memory by making three passes over the DVI file instead of two and only loading those characters actually used. Generally only useful on machines with a very limited amount of memory, like some PCs.

‘-A’ Print only the odd pages. This option uses TeX page numbers, not physical page numbers.

‘-b *num*’ Generate *num* copies of each page, but duplicating the page body rather than using the ‘/#copies’ PostScript variable. This can be useful in conjunction with a header file setting ‘bop-hook’ to do color separations or other neat tricks.

‘-B’ Print only the even pages. This option uses TeX page numbers, not physical page numbers.

‘-c *num*’ Generate *num* consecutive copies of every page, i.e., the output is uncollated. This merely sets the builtin PostScript variable ‘/#copies’.

'-C *num*' Generate *num* copies, but collated (by replicating the data in the PostScript file). Slower than the '-c' option, but easier on the hands, and faster than resubmitting the same PostScript file multiple times.

'-d *num*' Set the debug flags, showing what Dvips (thinks it) is doing. This will work unless Dvips has been compiled without the 'DEBUG' option (not recommended). See Section 2.4.1 [Debug options], page 5, for the possible values of *num*. Use '-d -1' as the first option for maximum output.

'-D *num*' Set both the horizontal and vertical resolution to *num*, given in dpi (dots per inch). This affects the choice of bitmap fonts that are loaded and also the positioning of letters in resident PostScript fonts. Must be between 10 and 10000. This affects both the horizontal and vertical resolution. If a high resolution (something greater than 400 dpi, say) is selected, the '-Z' flag should probably also be used. If you are using fonts made with Metafont, such as Computer Modern, 'mktexpk' needs to know about the value for *num* that you use or Metafont will fail. See the file `ftp://ftp.tug.org/tex/modes.mf` for a list of resolutions and mode names for most devices.

'-e *num*' Maximum drift in pixels of each character 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, while regaining the true position at the beginning of each new word, improves the spacing of letters in words.

'-E*' Generate an EPSF file with a tight bounding box. This only looks at marks made by characters and rules, not by any included graphics. In addition, it gets the glyph metrics from the TFM file, so characters that print outside their enclosing TFM box may confuse it. In addition, the bounding box might be a bit too loose if the character glyph has significant left or right side bearings. Nonetheless, this option works well enough for creating small EPSF files for equations or tables or the like. (Of course, Dvips output, especially when using bitmap fonts, is resolution-dependent and thus does not make very good EPSF files, especially if the images are to be scaled; use these EPSF files with care.) For multiple page input files, also specify '-i' to get each page as a separate EPSF file; otherwise, all the pages are overlaid in the single output file.

'-f*' Run as a filter. Read the DVI file from standard input and write the PostScript to standard output. The standard input must be seekable, so it cannot be a pipe. If your input must be a pipe, write a shell script that copies the pipe output to a temporary file and then points Dvips at this file. This option also disables the automatic reading of the `PRINTER` environment variable; use '-P$PRINTER' after the '-f' to read it anyway. It also 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 the '-f' to send it anyway.

'-F*' Write control-D (ASCII code 4) as the very last character of the PostScript file. This is useful when Dvips is driving the printer directly instead of working through a spooler, as is common on personal systems. On systems shared by more than one person, this is not recommended.

'-h *name*' Prepend *name* as an additional header file, or, if *name* is '-', suppress all header files. Any definitions in the header file get added to the PostScript 'userdict'.

'-i*' Make each section be a separate file; a *section* is a part of the document processed independently, most often created to avoid memory overflow. The filenames are created 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 '-i' is specified and '-S' is not, each page is output as a separate file. For instance, some phototypesetters cannot print more than ten or so consecutive pages before running out of steam; these options can be used to automatically split a book into ten-page sections, each to its own file.

'-j*' Download only needed characters from Type 1 fonts. This is the default in the current release. Some debugging flags trace this operation (see Section 2.4.1 [Debug options], page 5). You can also control partial downloading on a per-font basis (see Section 6.4 [psfonts.map], page 45).

'-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 a half inch in each dimension. It translates each page by a quarter inch and draws cross-style crop marks. It is mostly useful with typesetters that can set the page size automatically. This works by downloading 'crop.pro'.

'-K*' Remove comments in included PostScript graphics, font files, and headers; only necessary to get around bugs in spoolers or PostScript post-processing programs. Specifically, the '%%Page' comments, when left in, often cause difficulties. Use of this flag can cause other graphics to fail, however, since the PostScript header macros from some software packages read portion the input stream line by line, searching for a particular comment.

'-l [=]*num*'
The last page printed will be the first one numbered *num*. Default is the last page in the document. If *num* is prefixed by an equals sign, then it (and the argument to the '-p' option, if specified) is treated as a physical (absolute) page number, rather than a value to compare with the TeX '\count0' values stored in the DVI file. Thus, using '-l =9' will end with the ninth page of the document, no matter what the pages are actually numbered.

'-m*' Specify manual feed, if supported by the output device.

'-mode *mode*'
Use *mode* as the Metafont device name for path searching and font generation. This overrides any value from configuration files. With the default paths, explicitly specifying the mode also makes the program assume the fonts are in a subdirectory named *mode*. See section "TeX directory structure" in *Kpathsea*. If Metafont does not understand the *mode* name, see section "Unable to generate fonts" in *Kpathsea*.

'-M*' Turns off automatic font generation ('mktexpk'). If mktexpk, the invocation is appended to a file 'missfont.log' (by default) in the current directory. You can then execute the log file to create the missing files after fixing the problem. If the current directory is not writable and the environment variable or configuration file value 'TEXMFOUTPUT' is set, its value is used. Otherwise, nothing is written. The name 'missfont.log' is overridden by the 'MISSFONT_LOG' environment variable or configuration file value.

'-n *num*' Print at most *num* pages. Default is 100000.

'-N*' Turns off generation of structured comments such as '%%Page'; this may be necessary on some systems that try to interpret PostScript comments in weird ways, or on some PostScript printers. Old versions of TranScript in particular cannot handle modern Encapsulated PostScript. Beware: This also disables page movement, etc., in PostScript viewers such as Ghostview.

'-o *name*' Send output to the file *name*. If '-o' is specified without *name*, the default is '*file*.ps' where the input DVI file was '*file*.dvi'. If '-o' isn't given at all, the configuration file default is used.

If *name* is '-', output goes to standard output. If the first character of *name* is '!' or '|', then the remainder will be used as an argument to popen; thus, specifying '|lpr' as the output file will automatically queue the file for printing as usual. (The MS-DOS version will print to the local printer device 'PRN' when *name* is '|lpr' and a program by that name cannot be found.)

'-o' disables the automatic reading of the PRINTER environment variable, and turns off the automatic sending of control-D. See the '-f' option for how to override this.

'-O *x-offset*,*y-offset*'

Move the origin by *x-offset*,*y-offset*, a comma-separated pair of dimensions such as '.1in,-.3cm' (see Section 4.1 [papersize special], page 20). The origin of the page is shifted from the default position (of one inch down, one inch to the right from the upper left corner of the paper) by this amount. This is usually best specified in the printer-specific configuration file.

This is useful for a printer that consistently offsets output pages by a certain amount. You can use the file 'testpage.tex' to determine the correct value for your printer. Be sure to do several runs with the same O value—some printers vary widely from run to run.

If your printer offsets every other page consistently, instead of every page, your best recourse is to use 'bop-hook' (see Section 5.3.4 [PostScript hooks], page 30).

'-p [=]*num*'

The first page printed will be the first one numbered *num*. Default is the first page in the document. If *num* is prefixed by an equals sign, then it (and the argument to the '-l' option, if specified) is treated as a physical (absolute) page number, rather than a value to compare with the TeX '\count0' values stored in the DVI file. Thus, using '-p =3' will start with the third page of the document, no matter what the pages are actually numbered.

'-pp *first-last*'
: Print pages *first* through *last*; equivalent to '-p *first* -l *last*', except that multiple '-pp' options accumulate, unlike '-p' and '-l'. The '-' separator can also be ':'.

'-P *printer*'
: Read the configuration file 'config.*printer*' ('*printer*.cfg' on MS-DOS), which can set the output name (most likely 'o |lpr -P*printer*'), resolution, Metafont mode, and perhaps font paths and other printer-specific defaults. It works best to put sitewide defaults in the one master 'config.ps' file and only things that vary printer to printer in the 'config.*printer*' files; 'config.ps' is read before 'config.*printer*'.

  If no '-P' or '-o' is given, the environment variable PRINTER is checked. If that variable exists, and a corresponding 'config.*printer*' ('*printer*.cfg' on MS-DOS) file exists, it is read. See Section 3.4.1 [Configuration file searching], page 15.

'-q*'
: Run quietly. Don't chatter about pages converted, etc. to standard output; report no warnings (only errors) to standard error.

'-r*'
: Output pages in reverse order. By default, page 1 is output first.

'-R'
: Run securely. This disables shell command execution in \special (via '`', see Section 5.1.4 [Dynamic creation of graphics], page 27) and config files (via the 'E' option, see Section 3.4.2 [Configuration file commands], page 16), pipes as output files, and opening of any absolute filenames.

'-s*'
: Enclose the output in a global save/restore pair. This causes the file to not be truly conformant, and is thus not recommended, but is useful if you are driving a deficient printer directly and thus don't care too much about the portability of the output to other environments.

'-S *num*'
: Set the maximum number of pages in each 'section'. This option is most commonly used with the '-i' option; see its description above for more information.

'-t *papertype*'
: Set the paper type to *papertype*, usually defined in one of the configuration files, along with the appropriate PostScript code to select it (see Section 4.2 [Config file paper sizes], page 20). You can also specify a *papertype* of 'landscape', which rotates a document by 90 degrees. To rotate a document whose paper type is not the default, you can use the '-t' option twice, once for the paper type, and once for 'landscape'.

'-T *hsize,vsize*'
: Set the paper size to (*hsize,vsize*), a comma-separated pair of dimensions such as '.1in,-.3cm' (see Section 4.1 [papersize special], page 20). It overrides any paper size special in the DVI file.

'-U*'
: Disable a PostScript virtual memory-saving optimization that stores the character metric information in the same string that is used to store the bitmap information. This is only necessary when driving the Xerox 4045 PostScript interpreter, which has a bug that puts garbage on the bottom of each character. Not recommended unless you must drive this printer.

'-V*' Download non-resident PostScript fonts as bitmaps. This requires use of `mtpk` or `gsftopk` or `pstopk` or some combination thereof to generate the required bitmap fonts; these programs are supplied with Dvips. The bitmap must be put into '`psfonts.map`' as the downloadable file for that font. This is useful only for those fonts for which you do not have real outlines, being downloaded to printers that have no resident fonts, i.e., very rarely.

'-x *num*' Set the x magnification ratio to *num*/1000. Overrides the magnification specified in the DVI file. Must be between 10 and 100000. It is recommended that you use standard magstep values (1095, 1200, 1440, 1728, 2074, 2488, 2986, and so on) to help reduce the total number of PK files generated. *num* may be a real number, not an integer, for increased precision.

'-X *num*' Set the horizontal resolution in dots per inch to *num*.

'-y *num*' Set the y magnification ratio to *num*/1000. See '-x' above.

'-Y *num*' Set the vertical resolution in dots per inch to *num*.

'-z*' Pass '`html`' hyperdvi specials through to the output for eventual distillation into PDF. This is not enabled by default to avoid including the header files unnecessarily, and use of temporary files in creating the output. See Section 5.4 [Hypertext], page 31.

'-Z*' Compress bitmap fonts in the output file, thereby reducing the size of what gets downloaded. Especially useful at high resolutions or when very large fonts are used. May slow down printing, especially on early 68000-based PostScript printers. Generally recommend today, and can be enabled in the configuration file (see Section 3.4.2 [Configuration file commands], page 16).

3.3 Environment variables

Dvips looks for many environment variables, to define search paths and other things. The path variables are read as needed, after all configuration files are read, so they override values in the configuration files. (Except for `TEXCONFIG`, which defines where the configuration files themselves are found.)

See section "Path specifications" in *Kpathsea*, for details of interpretation of path and other environment variables common to all Kpathsea-using programs. Only the environment variables specific to Dvips are mentioned here.

`DVIPSFONTS`
: Default path to search for all fonts. Overrides all the font path config file options and other environment variables (see section "Supported file formats" in *Kpathsea*).

`DVIPSHEADERS`
: Default path to search for PostScript header files. Overrides the '`H`' config file option (see Section 3.4.2 [Configuration file commands], page 16).

`DVIPSMAKEPK`
: Overrides '`mktexpk`' as the name of the program to invoke to create missing PK fonts. You can change the arguments passed to the `mktexpk` program with

the `MAKETEXPK` environment variable; see section "MakeTeX script arguments" in *Kpathsea*.

`DVIPSRC` Specifies the name of the startup file (see Section 3.4.1 [Configuration file searching], page 15) which is read after '`config.ps`' but before any printer-specific configuration files.

`DVIPSSIZES`
Last-resort sizes for scaling of unfound fonts. Overrides the '`R`' definition in config files (see Section 3.4.2 [Configuration file commands], page 16).

`PRINTER` Determine the default printer configuration file. (Dvips itself does not use `PRINTER` to determine the output destination in any way.)

`TEXCONFIG`
Path to search for Dvips' '`config.`*printer*' configuration files, including the base '`config.ps`'. Using this single environment variable, you can override everything else. (The printer-specific configuration files are called '*printer*`.cfg`' on MS-DOS, but '`config.ps`' is called by that name on all platforms.)

`TEXPICTS` Path to search for included graphics files. Overrides the '`S`' config file option (see Section 3.4.2 [Configuration file commands], page 16). If not set, `TEXINPUTS` is looked for. See section "Supported file formats" in *Kpathsea*.

3.4 Dvips configuration files

This section describes in detail the Dvips-specific '`config.*`' device configuration files (called '`*.cfg`' on MS-DOS), which override the '`texmf.cnf`' configuration files generic to Kpathsea which Dvips also reads (see section "Config files" in *Kpathsea*).

For information about installing these files, including a prototype file you can copy, see Section 2.1 [config.ps installation], page 2.

3.4.1 Configuration file searching

The Dvips program loads many different configuration files, so that parameters can be set globally across the system, on a per-device basis, or individually by each user.

1. Dvips first reads (if it exists) '`config.ps`'; it is searched for along the path for Dvips configuration files, as described in section "Supported file formats" in *Kpathsea*.
2. A user-specific startup file is loaded, so individual users can override any options set in the global file. The environment variable `DVIPSRC`, if defined, is used as the specification of the startup file. If this variable is undefined, Dvips uses a platform-specific default name. On Unix Dvips looks for the default startup file under the name '`$HOME/.dvipsrc`', which is in the user's home directory. On MS-DOS and MS-Windows, where users generally don't have their private directories, the startup file is called '`dvips.ini`' and it is searched for along the path for Dvips configuration files (as described in section "Supported file formats" in *Kpathsea*.); users are expected to set this path as they see fit for their taste.

3. The command line is read and parsed: if the '-P*device*' option is encountered, at that point 'config.*device*' is loaded. Thus, the printer configuration file can override anything in the site-wide or user configuration file, and it can also override options in the command line up to the point that the '-P' option was encountered. (On MS-DOS, the printer configuration files are called '*device*.cfg', since DOS doesn't allow more than 3 characters after the dot in filenames.)
4. If no '-P' option was specified, and also the '-o' and '-f' command line options were not used, Dvips checks the environment variable PRINTER. If it exists, then 'config.*$PRINTER*' ('*$PRINTER*.cfg' on MS-DOS) is loaded (if it exists).

Because the '.dvipsrc' file is read before the printer-specific configuration files, individual users cannot override settings in the latter. On the other hand, the TEXCONFIG path usually includes the current directory, and can in any case be set to anything, so the users can always define their own printer-specific configuration files to be found before the system's.

A few command-line options are treated specially, in that they are not overridden by configuration files:

'-D' As well as setting the resolution, this unsets the mode, if the mode was previously set from a configuration file. If 'config.$PRINTER' is read, however, any 'D' or 'M' lines from there will take effect.

'-mode' This overrides any mode setting ('M' line) in configuration files. '-mode' does not affect the resolution.

'-o' This overrides any output setting ('o' line) in configuration files.

The purpose of these special cases is to (1) minimize the chance of having a mismatched mode and resolution (which 'mktexpk' cannot resolve), and (2) let command-line options override config files where possible.

3.4.2 Configuration file commands

Most of the configuration file commands are similar to corresponding command line options, but there are a few exceptions. When they are the same, we omit the description here.

As with command line options, many may be turned off by suffixing the letter with a zero ('0').

Within a configuration file, empty lines, and lines starting with a space, asterisk, equal sign, percent sign, or pound sign are ignored. There is no provision for continuation lines.

'@ *name hsize vsize*'
Define paper sizes. See Section 4.2 [Config file paper sizes], page 20.

'a*' Memory conservation. Same as '-a', see Section 3.2.2 [Option details], page 9.

'b *#copies*'
Multiple copies. Same as '-b', see Section 3.2.2 [Option details], page 9.

'D *dpi*' Output resolution. Same as '-D', see Section 3.2.2 [Option details], page 9.

'e *num*' Max drift. Same as '-e', see Section 3.2.2 [Option details], page 9.

'E *command*'
: Executes the command listed with `system`(3); can be used to get the current date into a header file for inclusion, for instance. Possibly dangerous; this may be disabled, in which case a warning will be printed if the option is used (and warnings are not suppressed).

'f*'
'F' Run as a filter. Same as '-f', see Section 3.2.2 [Option details], page 9.

'h *header*' Prepend *header* to output. Same as 'h-', see Section 3.2.2 [Option details], page 9.

'H *path*' Use *path* to search for PostScript header files. The environment variable `DVIPSHEADERS` overrides this.

'i *n*' Make multiple output files. Same as '-i -S *n*', see Section 3.2.2 [Option details], page 9.

'j*' Partially download Type 1 fonts. Same as '-j', see Section 3.2.2 [Option details], page 9.

'K*' Remove comments from included PostScript files. Same as '-K', see Section 3.2.2 [Option details], page 9.

'm *num*' Declare *num* as the memory available for fonts and strings in the printer. Default is 180000. This value must be accurate if memory conservation and document splitting is to work correctly. To determine this value, send the following file to the printer:

```
%! Hey, we're PostScript
/Times-Roman findfont 30 scalefont setfont 144 432 moveto
vmstatus exch sub 40 string cvs show pop showpage
```

The number printed by this file is the total memory free; it is usually best to tell Dvips that the printer has slightly less memory, because many programs download permanent macros that can reduce the memory in the printer. Some systems or printers can dynamically increase the memory available to a PostScript interpreter, in which case this file might return a ridiculously low number; for example, the NeXT computer and Ghostscript. In these cases, a value of one million works fine.

'M *mode*' Metafont mode. Same as '-mode', see Section 3.2.2 [Option details], page 9.

'N*' Disable structured comments. Beware: This also turns off displaying page numbers or changing to specific pagenumbers in PostScript viewers. Same as '-N', see Section 3.2.2 [Option details], page 9.

'o *name*' Send output to *name*. Same as '-', see Section 3.2.2 [Option details], page 9. In the file 'config.foo', a setting like this is probably appropriate:

```
o |lpr -Pfoo
```

The MS-DOS version will emulate spooling to `lpr` by printing to the local printer device 'PRN' if it doesn't find an executable program by that name in the current directory or along the `PATH`.

'`O` *xoff*`,`*yoff*'
: Origin offset. Same as '`-O`', see Section 3.2.2 [Option details], page 9.

'`p` [`+`]*name*'
: Examine *name* for PostScript font aliases. Default is '`psfonts.map`'. This option allows you to specify different resident fonts that different printers may have. If *name* starts with a '`+`' character, then the rest of the name (after any leading spaces) is used as an additional map file; thus, it is possible to have local map files pointed to by local configuration files that append to the global map file. This can be used for font families.

'`P` *path*'
: Use *path* to search for bitmap PK font files is *path*. The `PKFONTS`, `TEXPKS`, `GLYPHFONTS`, and `TEXFONTS` environment variables override this. See section "Supported file formats" in *Kpathsea*.

'`q*`'
'`Q`'
: Run quietly. Same as '`-q`', see Section 3.2.2 [Option details], page 9.

'`r*`'
: Page reversal. Same as '`-r`', see Section 3.2.2 [Option details], page 9.

'`R` *num1 num2* . . . '
: Define the list of default resolutions for PK fonts. If a font size actually used in a document is not available and cannot be created, Dvips will scale the font found at the closest of these resolutions to the requested size, using PostScript scaling. The resulting output may be ugly, and thus a warning is issued. To turn this last-resort scaling off, use a line with just the '`R`' and no numbers.

  The given numbers must be sorted in increasing order; any number smaller than the preceding one is ignored. This is because it is better to scale a font up than down; scaling down can obliterate small features in the character shape.

  The environment and config file values '`DVIPSSIZES`' or '`TEXSIZES`' override this configuration file setting.

  If no '`R`' settings or environment variables are specified, a list compiled in during installation is used. This default list is defined by the Makefile variable '`default_texsizes`', defined in the file '`make/paths.make`'.

'`s*`'
: Output global save/restore. Same as '`-s`', see Section 3.2.2 [Option details], page 9.

'`S` *path*'
: Use *path* to search for special illustrations (Encapsulated PostScript files or psfiles). The `TEXPICTS` and then `TEXINPUTS` environment variables override this.

'`T` *path*'
: Use *path* to search for TFM files. The `TFMFONTS` and then `TEXFONTS` environment variables overrides this. This path is used for resident fonts and fonts that can't otherwise be found.

'`U*`'
: Work around bug in Xerox 4045 printer. Same as '`-U`', see Section 3.2.2 [Option details], page 9.

'`V` *path*'
: Use *path* to search for virtual font files. This may be device-dependent if you use virtual fonts to simulate actual fonts on different devices.

'`W` [*string*]' If *string* is supplied, write it to standard error after reading all the configuration files; with no *string*, cancel any previous '`W`' message. This is useful in the default configuration file to remind users to specify a printer, for instance, or to notify users about special characteristics of a particular printer.

'`X` *num*' Horizontal resolution. Same as '`-X`', see Section 3.2.2 [Option details], page 9.

'`Y` *num*' Vertical resolution. Same as '`-Y`', see Section 3.2.2 [Option details], page 9.

'`Z*`' Compress bitmap fonts. Same as '`-Z`', see Section 3.2.2 [Option details], page 9.

4 Paper size and landscape orientation

Most TeX documents at a particular site are designed to use the standard paper size (letter size in the United States, A4 in Europe). The Dvips program can be customized either sitewide or for a particular printer.

But many documents are designed for other paper sizes. For instance, you may want to design a document that has the long edge of the paper horizontal. This can be useful when typesetting booklets, brochures, complex tables, or many other documents. This type of paper orientation is called *landscape* orientation (the default orientation is *portrait*). Alternatively, a document might be designed for ledger or A3 paper.

Since the intended paper size is a document design decision, not a printing decision, such information should be given in the TeX file and not on the Dvips command line. For this reason, Dvips supports a '`papersize`' special. It is hoped that this special will become standard over time for TeX previewers and other printer drivers.

4.1 '`papersize`' special

The format of the '`papersize`' special is

`\special{papersize=`*width*`,`*height*`}`

width is the horizontal size of the page, and *height* is the vertical size. The dimensions supported are the same as for TeX; namely, 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).

For a US letter size landscape document, the `papersize` would be:

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

An alternate specification of `landscape`:

```
\special{landscape}
```

This is supported for backward compatibility, but it is hoped that reventually the `papersize` comment will dominate.

Of course, such a `\special` only informs Dvips of the desired paper size; you must also adjust `\hsize` and `\vsize` in your TeX document typeset to those dimensions.

The `papersize` special must occur somewhere on the first page of the document.

4.2 Configuration file paper size command

The '`@`' command in a configuration file sets the paper size defaults and options. The first '`@`' command defines the default paper size. It has three possible parameters:

`@` [*name* [*hsize* *vsize*]]

If '`@`' is specified on a line by itself, with no parameters, it instructs Dvips to discard all previous paper size information (possibly from another configuration file).

If three parameters are given, with the first parameter being a name and the second and third being a dimension (as in '`8.5in`' or '`3.2cc`', just like in the '`papersize`' special), then the option is interpreted as starting a new paper size description, where *name* is the name and *hsize* and *vsize* define the horizontal and vertical size of the sheet of paper, respectively. For example:

```
@ letterSize 8.5in 11in
```

If both *hsize* and *vsize* are zero (you must still specify units!) then any page size will match. If the '`@`' character is immediately followed by a '`+`' sign, then the remainder of the line (after skipping any leading blanks) is treated as PostScript code to send to the printer, presumably to select that particular paper size:

```
@ letter 8.5in 11in
@+ %%BeginPaperSize: Letter
@+ letter
@+ %%EndPaperSize
```

After all that, if the first character of the line is an exclamation point, then the line is put in the initial comments section of the final output file; else, it is put in the setup section of the output file. For example:

```
@ legal 8.5in 14in
@+ ! %%DocumentPaperSizes: Legal
@+ %%BeginPaperSize: Legal
@+ legal
@+ %%EndPaperSize
```

When Dvips has a paper format name given on the command line, it looks for a match by the *name*; when it has a '`papersize`' special, it looks for a match by dimensions. The first match found (in the order the paper size information is found in the configuration file) is used. If nothing matches, a warning is printed and the first paper size is used. The dimensions must match within a quarter of an inch. Landscape mode for all paper sizes is automatically supported.

If your printer has a command to set a special paper size, then give dimensions of '`0in 0in`'; the PostScript code that sets the paper size can refer to the dimensions the user requested as '`hsize`' and '`vsize`'; these will be macros defined in the PostScript that return the requested size in default PostScript units. Virtually all of the PostScript commands you use here are device-dependent and degrade the portability of the file; that is why the default first paper size entry should not send any PostScript commands down (although a structured comment or two would be okay). Also, some printers want '`BeginPaperSize`' comments and paper size setting commands; others (such as the NeXT) want '`PaperSize`' comments and they will handle setting the paper size. There is no solution I could find that works for both (except maybe specifying both).

The Perl 5 script '`contrib/mkdvipspapers`' in the distribution directory may help in determining appropriate paper size definitions.

If your printers are configured to use A4 paper by default, the configuration file (probably the global '`config.ps`' in this case) should include this as the first '`@`' command:

```
@ A4size 210mm 297mm
@+ %%PaperSize: A4
```

so that `A4size` is used as the default, and not `A4` itself; thus, no PostScript `a4` command is added to the output file, unless the user explicitly says to use paper size '`a4`'. That is, by default, no paper size PostScript command should be put in the output, but Dvips will still know that the paper size is A4 because '`A4size`' is the first (and therefore default) size in the configuration file.

Executing the '`letter`' or '`a4`' or other PostScript operators cause the document to be nonconforming and can cause it not to print on certain printers, so the default paper size should not execute such an operator if at all possible.

4.3 Paper trays

Some printers, such as the Hewlett-Packard HP4si, have multiple paper trays. You can set up Dvips to take advantage of this using the `bop-hook` PostScript variable (see Section 5.3.4 [PostScript hooks], page 30).

For example, suppose you have an alternate tray stocked with letterhead paper; the usual tray has the usual paper. You have a document where you want the first page printed on letterhead, and the remaining pages on the usual paper. You can create a header file, say '`firstletterhead.PS`', with the following (PostScript) code (`bop-hook` is passed the current physical page number, which starts at zero):

`/bop-hook { dup 0 eq {` *alternatetray* `} {` *normaltray* `} ifelse } def`

where *alternatetray* and *normaltray* are the appropriate commands to select the paper trays. On the 4SI, *alternatetray* is '`statusdict begin 1 setpapertray end`' and *normaltray* is '`statusdict begin 0 setpapertray end`'.

Then, include the file with either

- the '`-h`' command-line option (see Section 3.2.2 [Option details], page 9); or
- the '`h`' config file option (see Section 3.4.2 [Configuration file commands], page 16); or
- '`\special{header=`*file*`}`' in your TeX document (see Section 5.2.1 [Including headers from TeX], page 28).

5 Interaction with PostScript

Dvips supports inclusion of PostScript figure files (e.g., Encapsulated PostScript), downloading other header files (e.g., fonts), including literal PostScript code, and hypertext.

5.1 PostScript figures

Scaling and including PostScript graphics is a breeze—if the PostScript file is correctly formed. Even if it is not, however, the file can usually be accommodated with just a little more work.

5.1.1 The bounding box comment

The most important feature of a good PostScript file from the standpoint of including it in another document is an accurate bounding box comment. Every well-formed PostScript file has a comment describing where on the page the graphic is located, and how big that graphic is.

This information is given as the lower left and upper right corners of the box just enclosing the graphic, and is thus referred to as the *bounding box*. These coordinates are given in the default PostScript units (there are precisely 72 PostScript units to the inch, like TEX big points) with respect to the lower left corner of the sheet of paper.

To see if a PostScript file has a bounding box comment, just look at the first few lines of the file. PostScript files are standard ASCII, so you can use any text editor to do this. If within the first few dozen lines there is a line like

```
%%BoundingBox: 25 50 400 300
```

(with any reasonable numbers), chances are very good that the file is Encapsulated PostScript and will work easily with Dvips. If the file contains instead a line like

```
%%BoundingBox: (atend)
```

the file is still probably Encapsulated PostScript, but the bounding box is given at the end of the file. Dvips needs it at the beginning. You can move it with that same text editor, or a simple script. (The bounding box is given in this way when the program that generated the PostScript couldn't know the size in advance, or was too lazy to compute it.)

If the document lacks a '`%%BoundingBox:`' altogether, you can determine one in a couple of ways. One is to use the '`bbfig`' program distributed with Dvips in the '`contrib`' directory. This can usually find the correct bounding box automatically; it works best with Ghostscript.

If the comment looks like this:

```
%%BoundingBox: 0 0 612 792
```

the graphic claims to take up an entire sheet of paper. This is usually a symptom of a bug in the program that generated it.

The other is to do it yourself: print the file. Now, take a ruler, and make the following measurements (in PostScript units, so measure in inches and multiply by 72): From the left edge of the paper to the leftmost mark on the paper is *llx*, the first number. From the bottom edge of the paper to the bottommost mark on the paper is *lly*, the second number. From the left edge of the paper to the rightmost mark on the paper is *urx*, the third number. The fourth and final number, *ury*, is the distance from the bottom of the page to the uppermost mark on the paper.

Once you have the numbers, add a comment of the following form as the second line of the document. (The first line should already be a line starting with the two characters '`%!`'; if it is not, the file probably isn't PostScript.)

`%%BoundingBox:` *llx lly urx ury*

Or, if you don't want to modify the file, you can simply write these numbers down in a convenient place and give them in your TeX document when you import the graphic, as described in the next section.

If the document does not have such a bounding box, or if the bounding box is given at the end of the document, or the bounding box is wrong, please complain to the authors of the software package that generated the file.

5.1.2 Using the EPSF macros

Once the figure file has a bounding box comment (see the previous section,) you are ready it the graphic into a TeX document. Many packages for using EPS files exist. One distributed with Dvips is the files '`epsf.tex`' (for plain TeX) and '`epsf.sty`' (for LaTeX). For plain TeX, add a line like this near the top of your input file:

```
\input epsf
```

If you are using LaTeX 2e, use the '`graphics`' or '`graphicx`' package. If you are using LaTeX 2.09, add the '`epsf`' style option, as in:

```
\documentstyle[12pt,epsf]{article}
```

In any case, the above only needs to be done once, no matter how many figures you plan to include.

Now, at the point you want to include a file, enter a line such as:

```
\epsffile{foo.eps}
```

If you are using LaTeX, you may need to add `\leavevmode` immediately before the `\epsffile` command to get certain environments to work correctly. If your file does not have a bounding box comment, you can supply the numbers as determined in the previous section, in the same order they would have been in a normal bounding box comment:

```
\epsffile[100 100 500 500]{foo.ps}
```

Now, save your changes and run TeX and Dvips; the output should have your graphic positioned at precisely the point you indicated, occupying the proper amount of space.

The `\epsffile` macro typesets the figure as a TeX `\vbox` at the point of the page that the command is executed. By default, the graphic will have its 'natural' width (namely, the width of its bounding box). The TeX box will have depth zero and its natural height. By default, the graphic will be scaled by any DVI magnification in effect, just as is everything else in your document. See the next section for more information on scaling.

If you want TeX to report the size of the figure as a message on your terminal when it processes each figure, give the command:

```
\epsfverbosetrue
```

5.1.2.1 EPSF scaling

Usually, you will want to scale an EPSF figure to some size appropriate for your document, since its natural size is determined by the creator of the EPS file.

The best way to do this is to assign the desired size to the TeX `\epsfxsize` or `\epsfysize` variables, whichever is more convenient for you. That is, put

`\epsfxsize=`*dimen*

right before the call to `\epsffile`. Then the width of the TeX box will be *dimen* and its height will be scaled proportionately. Similarly, you can set the vertical size with

`\epsfysize=`*dimen*

in which case the height will be set and the width scaled proportionally.

If you set both, both will be honored, but the aspect ratio of the included graphic may necessarily be distorted, i.e., its contents stretched in one direction or the other.

You can resize graphics in a more general way by redefining the `\epsfsize` macro. `\epsffile` calls this with two parameters: the natural horizontal and vertical sizes of the PostScript graphic. `\epsfsize` must expand to the desired horizontal size, that is, the width of the `\vbox`. Schematically:

`\def\epsfsize#1#2{`*body*`}`

Some useful definitions of *body*:

'`\epsfxsize`'
: This definition (the default) enables the default features listed above, by setting `\epsfxsize` to the same value it had before the macro was called.

'`#1`'
: Force the natural size by returning the first parameter (the original width).

'`0pt`'
: A special case, equivalent to '`#1`'.

'`0.5#1`'
: Scale to half the natural size.

'`\hsize`'
: Scale to the current `\hsize`. (In LaTeX, use `\textwidth` instead of `\hsize`.)

'`\ifnum#1>\hsize\hsize\else#1\fi`'
: If the natural width is greater than the current `\hsize`, scale to `\hsize`, otherwise use the natural width.

For compatibility with other PostScript drivers, it is possible to turn off the default scaling of included figures by the DVI magnification with the following TeX command:

```
\special{! /magscale false def}
```

Use of this command is not recommended because it will make the `\epsffile` graphics the "wrong" size if global magnification is being used, and it will cause any PostScript graphics to appear improperly scaled and out of position if a DVI to DVI program is used to scale or otherwise modify the document.

DVI magnification is not applied to any output from code you write in '`bop-hook`' or its ilk (see Section 5.3.4 [PostScript hooks], page 30),

5.1.2.2 EPSF clipping

By default, clipping is disabled for included EPSF images. This is because clipping to the bounding box dimensions often cuts off a small portion of the figure, due to slightly inaccurate bounding box arguments. The problem might be subtle; lines around the boundary of the image might be half their intended width, or the tops or bottoms of some text annotations might be sliced off. If you want to turn clipping on, just use the command

```
\epsfclipon
```

and to turn clipping back off, use

```
\epsfclipoff
```

5.1.3 'psfile' special

The basic special for file inclusion is as follows:

`\special{psfile=`*filename*`.ps` [*key*=*value*] `... }`

This downloads the PostScript file '*filename*`.ps`' such that the current point will be the origin of the PostScript coordinate system. The optional *key*=*value* assignments allow you to specify transformations on the PostScript.

The possible *key*s are:

'`hoffset`' The horizontal offset (default 0)

'`voffset`' The vertical offset (default 0)

'`hsize`' The horizontal clipping size (default 612)

'`vsize`' The vertical clipping size (default 792)

'`hscale`' The horizontal scaling factor (default 100)

'`vscale`' The vertical scaling factor (default 100)

'`angle`' The rotation (default 0)

'`clip`' Enable clipping to the bounding box

The dimension parameters are all given in PostScript units. The '`hscale`' and '`vscale`' are given in non-dimensioned percentage units, and the rotation value is specified in degrees. Thus

```
\special{psfile=foo.ps hoffset=72 hscale=90 vscale=90}
```

will shift the graphics produced by file '`foo.ps`' right by one inch and will draw it at 0.9 times normal size. Offsets are given relative to the point of the special command, and are unaffected by scaling or rotation. Rotation is counterclockwise about the origin. The order of operations is to rotate the figure, scale it, then offset it.

For compatibility with older PostScript drivers, it is possible to change the units that '`hscale`' and '`vscale`' are given in. This can be done by redefining '`@scaleunit`' in '`SDict`' by a TeX command such as

```
\special{! /@scaleunit 1 def}
```

The '`@scaleunit`' variable, which is by default 100, is what '`hscale`' and '`vscale`' are divided by to yield an absolute scale factor.

5.1.4 Dynamic creation of PostScript graphics files

PostScript is an excellent page description language—but it does tend to be rather verbose. Compressing PostScript graphics files can reduce them by factor of five or more. For this reason, if the name of an included PostScript file ends with '.Z' or '.gz', Dvips automatically runs 'gzip -d'. For example:

```
\epsffile[72 72 540 720]{foo.ps.gz}
```

Since the results of such a command are not accessible to TEX, if you use this facility with the 'epsf' macros, you need to supply the bounding box parameter yourself, as shown.

More generally, if the filename parameter to one of the graphics inclusion techniques starts with a left quote ('`'), the parameter is instead interpreted as a command to execute that will send the actual file to standard output. For example:

```
\special{psfile="`gnuplot foo"}
```

to include the file 'foo'. Of course, the command to be executed can be anything, including using a file conversion utility such as 'tek2ps' or whatever is appropriate. This feature can be disabled with the '-R' command-line option or 'R' configuration option.

5.1.5 Fonts in figures

You can use any font available to TEX and Dvips within a graphics file by putting a %*Font: line in the leading commentary of the file. Schematically, this looks like:

%*Font: *tfmname scaledbp designbp hex-start*:*hex-bitstring*

Here is the meaning of each of these elements:

tfmname The TEX TFM filename, e.g., 'cmr10'. You can give the same *tfmname* on more than one '%*Font' line; this is useful when the number of characters from the font used needs a longer *hex-bitstring* (see item below) than conveniently fits on one line.

scaledbp The size at which you are using the font, in PostScript points (TEX big points). $72\,\mathrm{bp} = 72.27\,\mathrm{pt} = 1\,\mathrm{in}$.

designbp The designsize of the font, again in PostScript points. This should match the value in the TFM file *tfmname*. Thus, for 'cmr10', it should be '9.96265'.

hex-start The character code of the first character used from the font, specified as two ASCII hexadecimal characters, e.g., '4b' or '4B' for 'K'.

hex-bitstring
An arbitrary number of ASCII hexadecimal digits specifying which characters following (and including) *hex-start* are used. This is treated as a bitmap. For example, if your figure used the single letter 'K', you would use '4b:8' for *hex-start* and *hex-bitstring*. If it used 'KLMNP', you would use '4b:f4'.

MetaPost's output figures contain lines like this for bitmap fonts used in a MetaPost label (see section "MetaPost" in *Web2c*).

5.2 PostScript header files

Header files are bits of PostScript included in the output file; generally they provide support for special features, rather than producing any printed output themselves. You can explicitly request downloading header files if necessary for some figure, or to achieve some special effect.

Dvips includes some headers on its own initiative, to implement features such as PostScript font reencoding, bitmap font downloading, handling of `\special`'s, and so on. These standard headers are the '`.pro`' files (for "prologue") in the installation directory '`$(psheaderdir)`'; they are created from the '`.lpro`' ("long prologue") files in the distribution by stripping comments, squeezing blank lines, etc., for maximum efficiency. If you want to peruse one of the standard header files, read the '`.lpro`' version.

The PostScript dictionary stack will be at the '`userdict`' level when header files are included.

5.2.1 Including headers from TEX

In order to get a particular graphic file to work, a certain font or header file might need to be sent first. The Dvips program provides support for this with the '`header`' `\special`. For instance, to ensure that '`foo.ps`' gets downloaded:

```
\special{header=foo.ps}
```

As another example, if you have some PostScript code that uses a PostScript font not built into your printer, you must download it to the printer. If the font isn't used elsewhere in the document, Dvips can't know you've used it, so you must include it in the same way, as in:

```
\special{header=putr.pfa}
```

to include the font definition file for Adobe Utopia Roman.

5.2.2 Including headers from the command line

You can include headers when you run Dvips, as well as from your document (see the previous section). To do this, run Dvips with the option '`-P` *header*'; this will read the file '`config.`*header*', which in turn can specify a header file to be downloaded with the '`h`' option. See Section 3.4.2 [Configuration file commands], page 16. These files are called '*header*`.cfg`' on MS-DOS.

You can arrange for the same file to serve as a '`-P`' config file and the downloadable header file, by starting the lines of PostScript code with a space, leaving only the '`h`' line and any comments starting in the first column. As an example, see '`contrib/volker/config.*`' ('`contrib/volker/*.cfg`' on MS-DOS). (These files also perform useful functions: controlling duplex/simplex mode on duplex printers, and setting various screen frequencies; '`contrib/volker/README`' explains further.)

5.2.3 Headers and memory usage

Dvips tries to avoid overflowing the printer's memory by splitting the output files into "sections" (see the '`-i`' option in Section 3.2.2 [Option details], page 9). Therefore, for all header files, Dvips debits the printer VM budget by some value. If the header file has, in its leading commentary a line of the form

```
%%VMusage: min max
```

then *max* is used. If there is no `%%VMusage` line, then the size (in bytes) of the header file is used as an approximation.

Illustrations (figure files) are also checked for `%%VMusage` line.

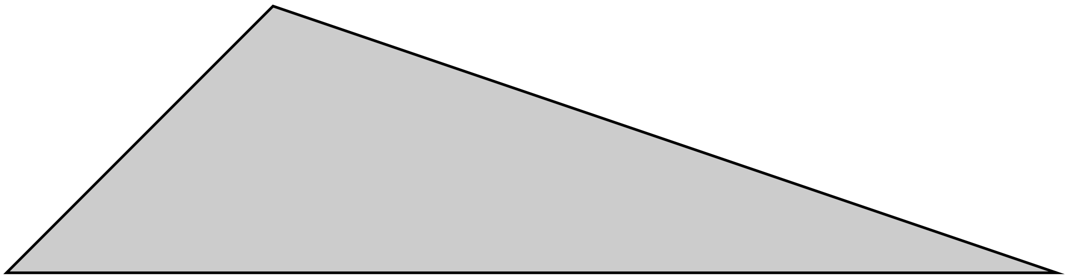
5.3 Literal PostScript

You can include literal PostScript code in your document in several ways.

5.3.1 " special: Literal PostScript

For simple graphics, or just for experimentation, literal PostScript code can be included. Simply use a `\special` beginning with a double quote character '`"`'; there is no matching closing '`"`'.

For instance, the following (simple) graphic:



was created by typing:

```
\vbox to 100bp{\vss % a bp is the same as a PostScript unit
  \special{" newpath 0 0 moveto 100 100 lineto 394 0 lineto
  closepath gsave 0.8 setgray fill grestore stroke}}
```

You are responsible for leaving space for such literal graphics, as with the `\vbox` above.

5.3.2 'ps' special

Generally, Dvips encloses specials in a PostScript save/restore pair, guaranteeing that the special will have no effect on the rest of the document. The '`ps`' special, however, allows you to insert literal PostScript instructions without this protective shield; you should understand what you're doing (and you shouldn't change the PostScript graphics state unless you are willing to take the consequences). This command can take many forms because it has had a torturous history; any of the following will work:

```
\special{ps:text}
\special{ps::text}
\special{ps::[begin]text}
\special{ps::[end]text}
```

(with longer forms taking precedence over shorter forms, when they are present). 'ps::' and 'ps::[end]' do no positioning, so they can be used to continue PostScript literals started with 'ps:' or 'ps::[begin]'.

In addition, the variant

```
\special{ps: plotfile filename}
```

inserts the contents of *filename* verbatim into the output (except for omitting lines that begin with %). An example of the proper use of literal specials can be found in the file 'rotate.tex', which makes it easy to typeset text turned in multiples of 90 degrees.

5.3.3 Literal headers: '!' \special

You can download literal PostScript header code in your TeX document, for use with (for example) literal graphics code that you include later. The text of a `\special` beginning with an '!' is copied into the output file. A dictionary `SDict` will be current when this code is executed; Dvips arranges for `SDict` to be first on the dictionary stack when any PostScript graphic is included, whether literally (the '"' special) or through macros (e.g., 'epsf.tex').

For example:

```
\special{! /reset { 0 0 moveto} def}
```

5.3.4 PostScript hooks

Besides including literal PostScript at a particular place in your document (as described in the previous section), you can also arrange to execute arbitrary PostScript code at particular times while the PostScript is printing.

If any of the PostScript names `bop-hook`, `eop-hook`, `start-hook`, or `end-hook` are defined in `userdict`, they will be executed at the beginning of a page, end of a page, start of the document, and end of a document, respectively.

When these macros are executed, the default PostScript coordinate system and origin is in effect. Such macros can be defined in headers added by the '-h' option or the 'header=' special, and might be useful for writing, for instance, 'DRAFT' across the entire page, or, with the aid of a shell script, dating the document. These macros are executed outside of the save/restore context of the individual pages, so it is possible for them to accumulate information, but if a document must be divided into sections because of memory constraints, such added information will be lost across section breaks.

The single argument to `bop-hook` is the physical page number; the first page gets zero, the second one, etc. `bop-hook` must leave this number on the stack. None of the other hooks are passed arguments.

As an example of what can be done, the following special will write a light grey 'DRAFT' across each page in the document:

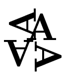
```
\special{!userdict begin /bop-hook{gsave 200 30 translate
65 rotate /Times-Roman findfont 216 scalefont setfont
0 0 moveto 0.7 setgray (DRAFT) show grestore}def end}
```

Using `bop-hook` or `eop-hook` to preserve information across pages breaks compliance with the Adobe document structuring conventions, so if you use any such tricks, you may also want to use the '-N' option to turn off structured comments (such as '%%Page'). Otherwise, programs that read your file will assume its pages are independent.

5.3.5 Literal examples

To finish off this section, the following examples of literal PostScript are presented without explanation:

```
\def\rotninety{\special{ps:currentpoint currentpoint translate 90
rotate neg exch neg exch translate}}\font\huge=cmbx10 at 14.4truept
\setbox0=\hbox to0pt{\huge A\hss}\vskip16truept\centerline{\copy0
\special{ps:gsave}\rotninety\copy0\rotninety\copy0\rotninety
\box0\special{ps:grestore}}\vskip16truept
```



```
\vbox to 2truein{\special{ps:gsave 0.3 setgray}\hrule height 2in
width\hsize\vskip-2in\special{ps:grestore}\font\big=cminch\big
\vss\special{ps:gsave 1 setgray}\vbox to 0pt{\vskip2pt
\line{\hss\hskip4pt NEAT\hss}\vss}\special{ps:0 setgray}%
\hbox{\raise2pt\line{\hss NEAT\hss}\special{ps:grestore}}\vss}
```

Some caveats are in order, however. Make sure that each `gsave` is matched with a `grestore` on the same page. Do not use `save` and `restore`; they can interact with the PostScript generated by Dvips if care is not taken. Try to understand what the above macros are doing before writing your own. The `\rotninety` macro especially has a useful trick that appears again and again.

5.4 HyperTEXt

Dvips has support for producing hypertext PostScript documents. If you specify the '-z' option, the 'html:' specials described below will be converted into 'pdfmark' PostScript operators to specify links. Without '-z', 'html:' specials are ignored.

The resulting PostScript can then be processed by a distiller program to make a PDF file. (It can still be handled by ordinary PostScript interpreters as well.) Various versions of both PC and Unix distillers are supported; Ghostscript includes limited distiller support (see Section 2.3 [Ghostscript installation], page 4).

Macros you can use in your TeX document to insert the specials in the first place are available from ‘*CTAN:*`/support/hypertex`’. For CTAN info, see section “unixtex.ftp” in *Kpathsea*.

This hypertext support (and original form of the documentation) was written by Mark Doyle and Tanmoy Bhattacharya as the ‘`dvihps`’ program. You can retrieve their software and additional documentation via the CTAN reference above. You may also be interested in the Java previewer IDVI, available at `http://www.win.tue.nl/~dickie/idvi`, and/or in `http://www.emrg.com/texpdf.html`, which describes the process of making PDF files from TeX files in more detail.

Mail archives for the original project are at `http://math.albany.edu:8800/hm/ht/`.

5.4.1 Hypertext caveats

If you intend to go all the way to PDF, you will probably want to use PostScript fonts exclusively, since the Adobe PDF readers are extremely slow when dealing with bitmap fonts. Commercial versions of the Computer Modern fonts are available from Blue Sky; public domain versions are available from CTAN sites (for CTAN info, see section “unixtex.ftp” in *Kpathsea*) in:

```
fonts/postscript/bakoma
fonts/postscript/paradissa
```

You may need to modify these fonts; see `http://xxx.lanl.gov/faq/bakoma.html`.

Also, the Adobe distillers prior to 2.1 drop trailing space characters (character code 32) from strings. Unfortunately, the PostScript fonts use this character code for characters other than space (notably the Greek letter psi in the `Symbol` font), and so these characters are dropped. This bug is fixed in version 2.1.

If you can't upgrade, One workaround is to change all the trailing blanks in strings to a character code that isn't in the font. This works because the default behavior is to substitute a blank for a missing character, i.e., the distiller is fooled into substituting the right character. For instance, with the Blue Sky fonts, you can globally replace ‘` )`’ with ‘`\200)`’ (with `sed`, for example) and get the desired result. With the public domain fonts, you will probably have to use a character code in the range 128 to 191 since these fonts duplicate the first 32 characters starting at 192 to avoid MS-DOS problems.

5.4.2 Hypertext specials

Current support for the World Wide Web in the TeX system does not involve modifying TeX itself. We need only define some specials; Arthur Smith (`apsmith@aps.org`), Tanmoy Bhattacharya, and Paul Ginsparg originally proposed and implemented the following:

```
html:<a href="xurl">
html:<a name="name">
html:</a>
html:<img src="xurl">
html:<base href="xurl">
```

Like all TeX `\special`'s, these produce no visible output, and are uninterpreted by TeX itself. They are instructions to DVI processors only.

Here, *xurl* is a standard WWW uniform resource locator (URL), possibly extended with a '#*type.string*' construct, where *type* is 'page', 'section', 'equation', 'reference' (for bibliographic references), 'figure', 'table', etc. For example,

```
\special{html:<a href="http://www.maths.tcd.ie/~tim/ch1.dvi#equation.1.1">}
```

is a link to equation (1.1) in an example document by Tim Murphy.

See http://www.w3.org/hypertext/WWW/Addressing/Addressing.html for a precise description of base URL's. (That itself is a URL, in case you were wondering.)

Descriptions of the \special's:

'href' Creates links in your TeX document. For example:

```
\special{html:<a href="http://www.tug.org/">}\TeX\ Users
Group\special{html:</a>}
```

The user will be able to click on the text 'TeX Users Group' while running Xdvi and get to the TUG home page. (By the way, this is for illustration. In practice, you most likely want to use macros to insert the \special commands; reference above.)

'name' Defines URL targets in your TeX documents, so links can be resolved. For example:

```
\special{html:<a name="#paradise">}Paradise\special{html:</a>}
is exactly where you are right now.
```

This will resolve an 'href="paradise"'.

'img' Links to an arbitrary external file. Interactively, a viewer is spawned to read the file according to the file extension and your 'mailcap' file (see the Xdvi documentation).

'base' Defines a base URL that is prepended to all the name targets. Typically unnecessary, as the name of the DVI file being read is used by default.

The 'img' and 'base' tags are not yet implemented in Dvips or the NeXTSTEP DVI viewer.

6 PostScript fonts

Dvips supports the use of PostScript fonts in TeX documents. To use a PostScript font conveniently, you need to prepare a corresponding virtual font; the program Afm2tfm, supplied with Dvips, helps with that.

All the necessary support for the standard 35 PostScript fonts (‘`AvantGarde-Book`’ through ‘`ZapfDingbats`’), plus other freely or commonly available PostScript fonts is available along with Dvips. To use these fonts, you need do nothing beyond what is mentioned in the installation procedure (see Chapter 2 [Installation], page 2). This chapter is therefore relevant only if you are installing new PostScript fonts not supplied with Dvips. (Or if you’re curious.)

6.1 Font concepts

The information needed to typeset using a particular font is contained in two files: a *metric file* that contains shape-independent information and a *glyph file* that contains the actual shapes of the font’s characters. A *virtual font* is an optional additional file that can specify special ways to construct the characters. TeX itself (or LaTeX) look only at the metric file, but DVI drivers such as Dvips look at all three of these files.

An *encoding file* defines the correspondence between the code numbers of the characters in a font and their descriptive names. Two encoding files used together can describe a reencoding that rearranges, i.e., renumbers, the characters of a font.

6.1.1 Metric files

A *metric file* describes properties of the font that are independent of what the characters actually look like. Aside from general information about the font itself, a metric file has two kinds of information: information about individual characters, organized by character code, and information about sequences of characters.

The per-character information specifies the width, height, depth, and italic correction of each character in the font. Any might be zero.

In addition to information on individual characters, the metric file specifies *kerning*, i.e., adding or removing space between particular character pairs. It further specifies *ligature* information: when a sequence of input characters should be typeset as a single (presumably different) “ligature” character. For example, it’s traditional for the input ‘`fi`’ to be typeset as ‘ﬁ’, not as ‘fi’ (with the dot of the ‘i’ colliding with ‘f’). (In English, the only common ligatures are ﬁ, ﬂ, ﬀ, ﬃ, and ﬄ.)

Different typesetting systems use different metric file formats:

- Each Postscript font has an *Adobe font metrics* (‘`.afm`’) file. These files are plain text, so you can inspect them easily. You can get AFM files for Adobe’s fonts from `ftp://ftp.adobe.com/pub/adobe/Fonts/AFMs`.
- TeX uses *TeX font metrics* (‘`.tfm`’) files. When you say ‘`\font =` *font*’ in your TeX document, TeX reads a file named ‘*font*`.tfm`’. (Well, except for the ‘`texfonts.map`’ feature; see section “Fontmap” in *Kpathsea*). TeX can then calculate the space occupied by characters from the font when typesetting. In addition, the DVI drivers you use to print or view the DVI file produced by TeX may need to look at the TFM file.

TFM files are binary (and hence are typically much smaller than AFM files). You can use the `tftopl` program (see section "tftopl invocation" in *Web2c*) that comes with TeX to transform a TFM file into a human-readable "property list" ('`.pl`') file. You can also edit a PL file and transform it back to a TeX-readable TFM with the companion program `pltotf` (see section "pltotf invocation" in *Web2c*). Editing metrics by hand is not something you're likely to want to do often, but the capability is there.

- ATM and other typesetting systems use *printer font metric* ('`.pfm`') files. These are binary files. They are irrelevant in the TeX world, and not freely available, so we will not discuss them further.

The Afm2tfm program distributed with Dvips converts an AFM file to a TFM file and performs other useful transformations as well. See Section 6.3 [Invoking afm2tfm], page 39.

6.1.2 Glyph files

Although a metric file (see the previous section) contains information about the spatial and other properties of the character at position 75, say, it contains nothing about what the character at position 75 actually looks like. The glyphs—the actual shapes of the letterforms in a font—are defined by other files, which we call glyph files. TeX itself only reads the TFM file for a font; it does not need to know character shapes.

A *glyph file* is a file that defines the shapes of the characters in a font. The shapes can be defined either by outlines or by bitmaps.

PostScript fonts are defined as *outline fonts*: Each character in the font is defined by giving the mathematical curves (lines, arcs, and splines) that define its contours. Different sizes of a character are generated by linearly scaling a single shape. For example, a 10-point 'A' is simply half the size of a 20-point 'A'. Nowadays, outline fonts usually also contain *hints*—additional information to improve the appearance of the font at small sizes or low resolutions.

Although various kinds of PostScript outline fonts exist, by far the most common, and the only one we will consider, is called *Type 1*. The glyph files for Postscript Type 1 fonts typically have names ending in '`.pfa`' ("printer font ASCII") or '`.pfb`' ("printer font binary").

In contrast, glyph files for Computer Modern and the other standard TeX fonts are *bitmap fonts*, generated from Metafont ('`.mf`') descriptions. The Metafont program distributed with TeX generates bitmaps from these descriptions.

The glyph files for TeX bitmap fonts are usually stored in *packed font* (PK) files. The names of these files end in '`.`*nnn*`pk`', where *nnn* is the resolution of the font in dots per inch. For example, '`cmr10.600pk`' contains the bitmaps for the '`cmr10`' font at a resolution of 600 dpi. (On DOS filesystems, it's more likely '`dpi600\cmr10.pk`'.)

Metafont actually outputs *generic font* (GF) files, e.g., '`cmr10.600gf`', but the GF files are usually converted immediately to PK format (using the `gftopk` utility that comes with TeX) since PK files are smaller and contain the same information. (The GF format is a historical artifact.)

6.1.3 Virtual fonts

A *virtual font* is constructed by extracting characters from one or more existing fonts and rearranging them, or synthesizing new characters in various ways. The explanation in this manual is intended to suffice for understanding enough about virtual fonts to use them with Dvips. It isn't a reference manual on virtual fonts. For more information: The primary document on virtual fonts is Donald E. Knuth, *TUGboat* 11(1), Apr. 1990, pp. 13–23, "Virtual Fonts: More Fun for Grand Wizards" ('*CTAN:*`/info/virtual-fonts.knuth`'; for CTAN info, see section "unixtex.ftp" in *Kpathsea*). (Don't be intimidated by the subtitle.)

A virtual font ('`.vf`') file specifies, for each character in the virtual font, a recipe for typesetting that character. A VF file, like a TFM file, is in a compressed binary format. The `vftovp` and `vptovf` programs convert a VF file to a human-readable VPL (virtual property list) format and back again. See section "vftovp invocation" in *Web2c*, and section "vptovf invocation" in *Web2c*.

In the case of a PostScript font *f* being used in a straightforward way, the recipe says: character *i* in the VF font is character *j* in font *f*. The font *f* is called a *base font*. For example, the VF file could remap the characters of the PostScript font to the positions where TeX expects to find them. See Section 6.1.4 [Encodings], page 36.

Since TeX reads only TFM files, not VF's, each VF must have a corresponding TFM for use with TeX. This corresponding TFM is created when you run `vptovf`.

You can *expand* virtual fonts into their base fonts with DVIcopy (see section "dvicopy invocation" in *Web2c*). This is useful if you are using a DVI translator that doesn't understand vf's itself.

6.1.4 Encodings

Every font, whatever its type, has an *encoding*, that specifies the correspondence between "logical" characters and character codes. For example, the ASCII encoding specifies that the character numbered 65 (decimal) is an uppercase 'A'. The encoding does not specify what the character at that position looks like; there are lots of ways to draw an 'A', and a glyph file (see Section 6.1.2 [Glyph files], page 35) tells how. Nor does it specify how much space that character occupies; that information is in a metric file (see Section 6.1.1 [Metric files], page 34).

TeX implicitly assumes a particular encoding for the fonts you use with it. For example, the plain TeX macro `\'`, which typesets an acute accent over the following letter, assumes the acute accent is at position 19 (decimal). This happens to be true of standard TeX fonts such as Computer Modern, as you might expect, but it is not true of normal PostScript fonts.

It's possible but painful to change all the macros that assume particular character positions. A better solution is to create a new font with the information for the acute accent at position 19, where TeX expects it to be. See Section 6.2 [Making a font available], page 38.

PostScript represents encodings as a sequence of 256 character names called an *encoding vector*. An *encoding file* ('`.enc`') gives such a vector, together with ligature and kerning information (with which we are not concerned at the moment). These encoding files are used by the Afm2tfm program. Encoding files are also downloaded to the PostScript interpreter in your printer if you use one of them in place of the default encoding vector for a particular PostScript font.

Examples of encodings: the '`dvips.enc`' encoding file that comes with Dvips in the '`reencode`' directory is a good (but not perfect) approximation to the TeX encoding for TeX's Computer Modern text fonts. This is the encoding of the fonts that originated with Dvips, such as '`ptmr.tfm`'. The distribution includes many other encoding files; for example, '`8r.enc`', which is the base font for the current PostScript font distribution, and three corresponding to the TeX mathematics fonts: '`texmext.enc`' for math extensions, '`texmital.enc`' for math italics, and '`texmsym.enc`' for math symbols.

6.1.5 How PostScript typesets a character

The output of Dvips is a program in the PostScript language that instructs your (presumably PostScript-capable) printer how to typeset your document by transforming it into toner on paper. Your printer, in turn, contains a PostScript interpreter that carries out the instructions in this typesetting program.

The program must include the definition of any PostScript fonts that you use in your document. Fonts built into your printer (probably the standard 35: '`Times-Roman`', '`ZapfDingbats`', ...) are defined within the interpreter itself. Other fonts must be downloaded as pfa or pfb files (see Section 6.1.2 [Glyph files], page 35) from your host (the computer on which you're running Dvips).

You may be wondering exactly how a PostScript interpreter figures out what character to typeset, with this mass of metrics, glyphs, encodings, and other information. (If you're not wondering, skip this section ...)

The basic PostScript operator for imaging characters is `show`. Suppose you've asked TeX to typeset an 'S'. This will eventually wind up in the Dvips output as the equivalent of this PostScript operation:

```
(S) show
```

Here is how PostScript typesets the 'S':

1. PostScript interpreters use ASCII; therefore 'S' is represented as the integer 83. (Any of the 256 possible characters representable in a standard 8-bit byte can be typeset.)
2. A PostScript *dictionary* is a mapping of names to arbitrary values. A font, to the interpreter, is a dictionary which contains entries for certain names. (If these entries are missing, the interpreter refuses to do anything with that font.)

   PostScript has a notion of "the current font"—whatever font is currently being typeset in.
3. One of the mandatory entries in a font dictionary is '`Encoding`', which defines the encoding vector (see Section 6.1.4 [Encodings], page 36) for that font. This vector of 256 names maps each possible input character to a name.
4. The interpreter retrieves the entry at position 83 of the encoding vector. This value is a PostScript name: `/S`.
5. For Type 1 fonts (we're not going to discuss anything else), the interpreter now looks up `/S` as a key in a dictionary named `CharStrings`, another mandatory entry in a font dictionary.

6. The value of `S` in `CharStrings` is the equivalent of a series of standard PostScript commands like '`curveto`', '`lineto`', '`fill`', and so on. These commands are executed to draw the character. There can also be *hint information* that helps adapt the character to low-resolution rasters. (See Section 6.1.2 [Glyph files], page 35.) The commands are actually represented in a more compact way than standard PostScript source; see the Type 1 book for details.

This method for typesetting characters is used in both Level 1 and Level 2 PostScript. See the PostScript reference manuals for more information.

6.2 Making a PostScript font available

To make a PostScript font available in a TeX document, you need to install the font on your system and then define it within the document. Once you have installed the font, of course, it is available for any document thereafter and you don't need to reinstall it. You must have an AFM file for any font you install. Unless the font is built into your printer, you must also have a PFA or PFB file.

In the following examples, we use the font '`Times-Roman`' to illustrate the process. But you should use the prebuilt fonts for Times and the other standard fonts, rather than rebuilding them. The prebuilt fonts are made using a more complicated process than that described here, to make them work as well as possible with TeX. So following the steps in this manual will not generate files identical to the distributed ones. See Section 2.2 [PostScript font installation], page 3, for pointers to the prebuilt fonts.

Installation of a PostScript font proceeds in three steps. See Section 6.1 [Font concepts], page 34, for descriptions of the various files involved.

1. Run `afm2tfm` to create a TFM file for the original font, and the VPL form of the virtual font:

   ```
   afm2tfm Times-Roman -v ptmr rptmr
   ```

2. Run `vptovf` to generate a VF and TFM file for the virtual font from the VPL file:

   ```
   vptovf ptmr.vpl ptmr.vf ptmr.tfm
   ```

3. Insert an entry for the font in '`psfonts.map`' (See Section 6.4 [psfonts.map], page 45):

   ```
   rptmr      Times-Roman                 <ptmr8a.pfa
   ```

4. Install the files in the standard locations, as in:

   ```
   cp ptmr.vf fontdir/vf/...
   cp *ptmr.tfm fontdir/tfm/...
   cp ptmr.afm fontdir/afm/...
   cp ptmr.pf? fontdir/type1/...
   ```

The simplest invocation of Afm2tfm to make virtual fonts goes something like this:

```
afm2tfm Times-Roman -v ptmr rptmr
```

This reads the file '`Times-Roman.afm`', and produces two files as output, namely the virtual property list'file '`ptmr.vpl`', and the "raw" font metric file '`rptmr.tfm`'. To use the font in TeX, you first run

```
vptovf ptmr.vpl ptmr.vf ptmr.tfm
```

You should then install the virtual font file '`ptmr.vf`' where Dvips will see it and install '`ptmr.tfm`' and '`rptmr.tfm`' where TeX and Dvips will see them.

Using these raw fonts is not recommended; there are no raw fonts in the prebuilt PostScript fonts distributed along with Dvips. But nevertheless, that's how Afm2tfm presently operates, so that's what we document here. The '`r`' prefix convention is likewise historical accident.

You can also make more complex virtual fonts by editing '`ptmr.vpl`' before running '`vptovf`'; such editing might add the uppercase Greek characters in the standard TeX positions, for instance. (This has already been done for the prebuilt fonts.)

Once the files have been installed, you're all set. You can now do things like this in TeX:

```
\font\myfont = ptmr at 12pt
\myfont Hello, I am being typeset in 12-point Times-Roman.
```

Thus, we have two fonts, one actual ('`rptmr`', which is analogous to the font in the printer) and one virtual ('`ptmr`', which has been remapped to the standard TeX encoding (almost)), and has typesetting know-how added. You could also say

```
\font\raw = rptmr at 10pt
```

and typeset directly with that, but then you would have no ligatures or kerning, and you would have to use Adobe character positions for special letters like Æ. The virtual font '`ptmr`' not only has ligatures and kerning, and most of the standard accent conventions of TeX, it also has a few additional features not present in the Computer Modern fonts. For example, it includes all the Adobe characters (such as the Polish ogonek and the French guillemots). The only things you lose from ordinary TeX text fonts are the dotless '`j`' (which can be hacked into the VPL file with literal PostScript specials if you have the patience) and uppercase Greek letters (which just don't exist unless you buy them separately). See Section 6.3.1.4 [Reencoding with Afm2tfm], page 41.

As a final step you need to record information about both the virtual font and the original font (if you ever might want to use it) in the '`psfonts.map`' file (see Section 6.4 [psfonts.map], page 45). For our example, you'd insert the following into '`psfonts.map`':

```
rptmr     Times-Roman                 <ptmr8a.pfa
```

Of course, `Times-Roman` is already built in to most every printer, so there's no need to download any Type 1 file for it. But if you are actually following these instructions for new fonts, most likely they are not built in to the printer.

These PostScript fonts can be scaled to any size. Go wild! Using PostScript fonts, however, does use up a great deal of the printer's memory and it does take time. You may find downloading bitmap fonts (possibly compressed, with the '`Z`' option) to be faster than using the built-in PostScript fonts.

6.3 Invoking Afm2tfm

The Afm2tfm program converts an AFM file for a PostScript font to a TFM file and a VPL file for a corresponding virtual font (or, in its simplest form, to a TFM file for the PostScript font itself). The results of the conversion are affected by the command-line options and especially by the reencodings you can specify with those options. You can also obtain special effects such as an oblique font.

An alternative to Afm2tfm for creating virtual fonts is Alan Jeffrey's `fontinst` program, available from '*CTAN:*`fonts/utilities/fontinst`' (for CTAN info, see section "unixtex.ftp" in *Kpathsea*).

6.3.1 Changing font encodings

Afm2tfm allows you to specify a different encoding for a PostScript font (for a general introduction to encodings, see Section 6.1.4 [Encodings], page 36). The '`-t`' options changes the TeX encoding, '`-p`' changes the PostScript encoding, and '`-T`' changes both simultaneously, as detailed in the sections below.

6.3.1.1 '`-t`': Changing TeX encodings

To build a virtual font with Afm2tfm, you specify the '`-v`' or '`-V`' option. You can then specify an encoding for that virtual font with '`-t` *tex-enc*'. ('`-t`' is ignored if neither '`-v`' nor '`-V`' is present.) Any ligature and kerning information you specify in *tex-enc* will be used in the VPL, in addition to the ligature and kerning information from the AFM file.

If the AFM file has no entry for a character specified in *tex-enc*, that character will be omitted from the output VPL.

The '`-t`' option is likely to be needed when you have a PostScript font corresponding to a TeX font other than a normal text font such as Computer Modern. For instance, if you have a PostScript font that contains math symbols, you'd probably want to use the encoding in the '`texmsym.enc`' file supplied with Dvips. (For a start; to actually get usable math fonts, you have to define much more than just an encoding.)

6.3.1.2 '`-p`': Changing PostScript encodings

By default, Afm2tfm uses the encoding it finds in the AFM file. You can specify a different PostScript encoding with '`-p` *ps-enc*'. This makes the raw TFM file (the one output by Afm2tfm) have the encoding specified in the encoding file *ps-enc*. Any ligature or kern information specified in *ps-enc* is ignored by Afm2tfm, since ligkern info is always omitted from the raw TFM.

If you use this option, you must also arrange to download *ps-enc* as part of any document that uses this font. You do this by adding a line like the following one to '`psfonts.map`' (see Section 6.4 [psfonts.map], page 45):

```
zpopr Optima "MyEncoding ReEncodeFont" <myenc.enc
```

Using '`-p`' is the only way to access characters in a PostScript font that are neither encoded in the AFM file nor constructed from other characters. For instance, Adobe's '`Times-Roman`' font contains the extra characters '`trademark`' and '`registered`' (among others); these can only be accessed through such a PostScript reencoding.

In fact, the '8r' base encoding used for the current PostScript font distribution (available at `ftp://ftp.tug.org/tex/psfonts.tar.gz`) does do this reencoding, for precisely this reason.

6.3.1.3 '`-T`': Changing both TeX and PostScript encodings

The option '`-T` *enc-file*' is equivalent to '`-p` *enc-file* `-t` *enc-file*'. If you make regular use of a private non-standard reencoding '`-T`' is usually a better idea than the individual options, to avoid unexpected inconsistencies in mapping otherwise. An example of when you might use this option is a dingbats font: when you have a TeX encoding that is designed to be used with a particular PostScript font.

6.3.1.4 Reencoding with Afm2tfm

The Afm2tfm program creates the TFM and VF files for the virtual font corresponding to a PostScript font by *reencoding* the PostScript font. Afm2tfm generates these files from two encodings: one for TeX and one for PostScript. The TeX encoding is used to map character numbers to character names while the PostScript encoding is used to map each character name to a possibly different number. In combination, you can get access to any character of a PostScript font at any position for TeX typesetting.

In the default case, when you specify none of the `-t`, `-p`, or `-T` options, Afm2tfm uses a default TeX encoding (which mostly corresponds to the Computer Modern text fonts) and the PostScript encoding found in the AFM file being read. The reencoding is also sometimes called a *remapping*.

For example, the default encodings reencode the acute accent in two steps: first the default TeX encoding maps the number 19 to the character name '`acute`'; then the default PostScript encoding, as found in the AFM file for an ordinary PostScript font, maps the character name '`acute`' to the number 194. (The PostScript encoding works in reverse, by looking in the encoding vector for the name and then yielding the corresponding number.) The combined mapping of 19 to 194 shows up explicitly in the VF file and also implicitly in the fact that the properties of PostScript character 194 appear in position 19 of the TFM file for the virtual font.

The default encoding of the distributed fonts (e.g., '`ptmr.tfm`') mostly follows plain TeX conventions for accents. The exceptions: the Hungarian umlaut (which is at position `0x7D` in '`cmr10`', but position `0xCD` in '`ptmr`'); the dot accent (at positions `0x5F` and `0xC7`, respectively); and the Scandinavian A ring `\AA`, whose definition needs different tweaking. In order to use these accents with PostScript fonts or in math mode when `\textfont0` is a PostScript font, you will need to use the following definitions. These definitions will not work with the Computer Modern fonts for the relevant accents. They are already part of the distributed '`psfonts.sty`' for use with LaTeX.

```
\def\H#1{{\accent"CD #1}}
\def\.#1{{\accent"C7 #1}}
\def\dot{\mathaccent"70C7 }
\newdimen\aadimen
\def\AA{\leavevmode\setbox0\hbox{h}\aadimen\ht0
  \advance\aadimen-1ex\setbox0\hbox{A}\rlap{\raise.67\aadimen
  \hbox to \wd0{\hss\char'27\hss}}A}
```

As a kind of summary, here are the '`CODINGSCHEME`'s that result from the various possible choices for reencoding.

default encoding
: `(CODINGSCHEME TeX text + AdobeStandardEncoding)`

'`-p dc.enc`'
: `(CODINGSCHEME TeX text + DCEncoding)`

'`-t dc.enc`'
: `(CODINGSCHEME DCEncoding + AdobeStandardEncoding)`

'`-T dc.enc`'
: `(CODINGSCHEME DCEncoding + DCEncoding)`

The 'CODINGSCHEME' line appears in the VPL file but is ignored by Dvips.

6.3.1.5 Encoding file format

Afm2tfm's encoding files have the same format as an encoding vector in a PostScript font. Here is a skeletal example:

```
% Comments are ignored, unless the first word after the percent sign
% is 'LIGKERN'; see below.
/MyEncoding [ % exactly 256 entries follow, each with a leading '/'
  /Alpha /Beta /Gamma /Delta ...
  /A /B ... /Z
  ...  /.notdef /xfooaccent /yfooaccent /zfooaccent
] def
```

These encoding files are downloaded as part of changing the encoding at the PostScript level (see the previous section).

Comments, which start with a percent sign and continue until the end of the line, are ignored unless they start with 'LIGKERN' (see below).

The first non-comment word of the file must start with a forward slash '/' (i.e., a PostScript literal name) and defines the name of the encoding. The next word must be an left bracket '['. Following that must be precisely 256 character names; use '/.notdef' for any that you want to leave undefined. Then there must be a matching right bracket]. A final 'def' token is optional. All names are case-sensitive.

Any ligature or kern information is given as a comment. If the first word after the '%' is 'LIGKERN', then the entire rest of the line is parsed for ligature and kern information. This ligature and kern information is given in groups of words: each group is terminated by a space and a semicolon and (unless the semicolon is at the end of a line) another space.

In these LIGKERN statements, three types of information may be specified. These three types are ligature pairs, kerns to ignore, and the character value of this font's boundary character.

Throughout a LIGKERN statement, the boundary character is specified as '||'. To set the font's boundary character value for TeX:

```
% LIGKERN || = 39 ;
```

To indicate a kern to remove, give the names of the two characters (without the leading slash) separated by '{}', as in 'one {} one ;'. This is intended to be reminiscent of the way you might use '{}' in a TeX file to turn off ligatures or kerns at a particular location. Either or both of the character names can be given as '*', which is a wild card matching any character; thus, all kerns can be removed with '* {} * ;'.

To specify a ligature, specify the names of the pair of characters, followed by the ligature operation (as in Metafont), followed by the replacing character name. Either (but not both) of the first two characters can be '||' to indicate a word boundary.

The most common operation is '=:' meaning that both characters are removed and replaced by the third character, but by adding the '|' character on either side of the '=:', you can retain either or both of the two leading characters. In addition, by suffixing the ligature operation with one or two '>' signs, you can make the ligature scanning operation skip that many resulting characters before proceeding. This works just like in Metafont.

For example, the 'fi' ligature is specified with '`f i =: fi ;`'. A more convoluted ligature is '`one one |=:|>> exclam ;`' which separates a pair of adjacent `1`'s with an exclamation point, and then skips over two of the resulting characters before continuing searching for ligatures and kerns. You cannot give more `>`'s than `|`'s in an ligature operation, so there are a total of eight possibilities:

```
=: |=: |=:> =:| =:|> |=:| |=:|> |=:|>>
```

The default set of ligatures and kerns built in to Afm2tfm is:

```
% LIGKERN question quoteleft =: questiondown ;
% LIGKERN exclam quoteleft =: exclamdown ;
% LIGKERN hyphen hyphen =: endash ; endash hyphen =: emdash ;
% LIGKERN quoteleft quoteleft =: quotedblleft ;
% LIGKERN quoteright quoteright =: quotedblright ;
% LIGKERN space {} * ; * {} space ; 0 {} * ; * {} 0 ;
% LIGKERN 1 {} * ; * {} 1 ; 2 {} * ; * {} 2 ; 3 {} * ; * {} 3 ;
% LIGKERN 4 {} * ; * {} 4 ; 5 {} * ; * {} 5 ; 6 {} * ; * {} 6 ;
% LIGKERN 7 {} * ; * {} 7 ; 8 {} * ; * {} 8 ; 9 {} * ; * {} 9 ;
```

6.3.2 Special font effects

Besides the reencodings described in the previous section, Afm2tfm can do other manipulations. (Again, it's best to use the prebuilt fonts rather than attempting to remake them.)

'`-s` *slant*' makes an obliqued variant, as in:

```
afm2tfm Times-Roman -s .167 -v ptmro rptmro
```

This creates '`ptmro.vpl`' and '`rptmro.tfm`'. To use this font, put the line

```
rptmro Times-Roman ".167 SlantFont"
```

into '`psfonts.map`'. Then '`rptmro`' (our name for the obliqued Times) will act as if it were a resident font, although it is actually constructed from Times-Roman via the PostScript routine `SlantFont` (which will slant everything 1/6 to the right, in this case).

Similarly, you can get an expanded font with

```
afm2tfm Times-Roman -e 1.2 -v ptmrre rptmrre
```

and by recording the pseudo-resident font

```
rptmrre Times-Roman "1.2 ExtendFont"
```

in '`psfonts.map`'.

You can also create a small caps font with a command such as

```
afm2tfm Times-Roman -V ptmrc rptmrc
```

This will generate a set of pseudo-small caps mapped into the usual lowercase positions and scaled down to 0.8 of the normal cap dimensions. You can also specify the scaling as something other than the default 0.8:

```
afm2tfm Times-Roman -c 0.7 -V ptmrc rptmrc
```

It is unfortunately not possible to increase the width of the small caps independently of the rest of the font. If you want a really professional looking set of small caps, you need to acquire a small caps font.

To change the `PaintType` in a font from filled (0) to outlined (2), you can add `"/PaintType 2 store"` to '`psfonts.map`', as in the following:

```
rphvrl   Helvetica "/PaintType 2 store"
```

Afm2tfm writes to standard output the line you need to add to '`psfonts.map`' to use that font, assuming the font is resident in the printer; if the font is not resident, you must add the '`<`*filename*' command to download the font. Each identical line only needs to be specified once in the '`psfonts.map`' file, even though many different fonts (small caps variants, or ones with different output encodings) may be based on it.

6.3.3 Afm2tfm options

Synopsis:

`afm2tfm [`*option*`]... `*afmfile*`[.afm] [`*tfmfile*`[.tfm]]`

Afm2tfm reads *afmfile* and writes a corresponding (but raw) TFM file. If *tfmfile* is not supplied, the base name of the AFM file is extended with '`.tfm`' to get the output filename.

The simplest example:

```
afm2tfm Times-Roman rptmr
```

The TFM file thus created is *raw* because it omits ligature and kern information, and does no character remapping; it simply contains the character information in the AFM file in TFM form, which is the form that TeX understands. The characters have the same code in the TFM file as in the AFM file. For text fonts, this means printable ASCII characters will work ok, but little else, because standard PostScript fonts have a different encoding scheme than the one that plain TeX expects (see Section 6.1.4 [Encodings], page 36). Although both schemes agree for the printable ASCII characters, other characters such as ligatures and accents vary. Thus, in practice, it's almost always desirable to create a virtual font as well with the '`-v`' or '`-V`' option. See Section 6.2 [Making a font available], page 38.

The command line options to Afm2tfm:

'`-c` *ratio*'
: See '`-V`'; overrides the default ratio of 0.8 for the scaling of small caps.

'`-e` *ratio*'
: Stretch characters horizontally by *ratio*; if less than 1.0, you get a condensed font.

'`-O`'
: Output all character codes in the '`vpl`' file as octal numbers, not names; this is useful for symbol or other special-purpose fonts where character names such as '`A`' have no meaning.

'`-p` *ps-enc*'
: Use *ps-enc* for the destination (PostScript) encoding of the font; *ps-enc* must be mentioned as a header file for the font in '`psfonts.map`'. See Section 6.3.1.2 [Changing PostScript encodings], page 40.

'`-s` *slant*'
: Slant characters to the right by *slant*. If *slant* is negative, the letters slope to the left (or they might be upright if you start with an italic font).

'`-t` *tex-enc*'
: Use *tex-enc* for the target (TeX) encoding of the font. Ligature and kern information may also be specified in *file*. *file* is not mentioned in '`psfonts.map`'.

'`-T` *ps-tex-enc*'
: Use *ps-tex-enc* for both the PostScript and target TeX encodings of the font. Equivalent to '`-p` *file* `-t` *file*'.

‘`-u`’
: Use only those characters specified in the TEX encoding, and no others. By default, Afm2tfm tries to include all characters in the input font, even those not present in the TEX encoding (it puts them into otherwise-unused positions, arbitrarily).

‘`-v` *vpl-file*’
: Output a VPL (virtual property list) file, as well as a TFM file.

‘`-V` *vpl-file*’
: Same as ‘`-v`’, but the virtual font generated is a pseudo small caps font obtained by scaling uppercase letters by 0.8 to typeset lowercase. This font handles accented letters and retains proper kerning.

6.4 ‘`psfonts.map`’: PostScript font catalog

The ‘`psfonts.map`’ file associates a PostScript font with related files and constructs. Each line has the format:

filename PostScript-name options

For example, the line

```
rpstrn StoneInformal <StoneInformal.pfb
```

causes Dvips to download ‘`StoneInformal.pfb`’ if your document (just as if it were a header file, see Section 5.2 [Header files], page 28) uses the font ‘`StoneInformal`’.

If the ‘`j`’ config file or command-line option is enabled, ‘`StoneInformal.pfb`’ will be *partially downloaded*—only those characters your document actually uses will be extracted and downloaded, and the remainder discarded. See Section 3.2.2 [Option details], page 9.

Filenames of fonts that are partially downloaded are surrounded by curly braces (‘`{...}`’) in the progress report Dvips writes to standard output. Wholly-downloaded fonts appear inside angle brackets (‘`<...>`’), like other downloaded files.

Adobe Multiple Master fonts, such as Minion, cannot be partially downloaded. To partially download in general, but avoid partial downloading for individual fonts, use ‘`<<`’ instead ‘`<`’:

```
pmnr8r Minion <<Minion.pfb
```

You can generate transformed fonts with a line like this:

```
rpstrc StoneInformal <StoneInformal.pfb ".8 ExtendFont"
```

See Section 6.3.2 [Special font effects], page 43, for a complete list of font effects.

You can change the encoding of the Type 1 font at the PostScript level with a ‘`ReEncodeFont`’ instruction, plus the name of the encoding file. This allows you access to characters that may be present in the Type 1 font file, but not encoded by default—most of the preaccented characters, for example. An example:

```
pstrn8r StoneInformal "TeXBase1Encoding ReEncodeFont" <8r.enc <pstrn8a.pfb
```

The ‘`8r`’ encoding mentioned here has been designed to serve as a base for all downloadable fonts; it allows access to all the characters commonly present in a Type 1 font. For more details, see the ‘`8r.enc`’ source file that comes with (and is installed with) Dvips.

You may notice that the same syntax is used for downloading encoding vectors and Type 1 font files. To make your intentions clear, you can also use ‘`<[`’ to explicitly indicate you are downloading an encoding vector, as in:

```
pstrn8r StoneInformal "TeXBase1Encoding ReEncodeFont" <[8r.enc <pstrn8a.pfb
```

If the filename of your encoding vector does not end in '.enc', and you are using partial font downloading, you must use the '<[' syntax, or Dvips will not download the font properly.

Similarly, the name of the Type 1 font file itself must have extension '.pfa' or '.pfb' for partial downloading to work properly.

When using PFB files, Dvips is smart enough to unpack the binary PFB format into printable ASCII so there is no need to perform this conversion yourself. In addition, Dvips scans the font to determine its memory usage, just as it does for other header files (see Section 5.2 [Header files], page 28).

Here is a brief summary of how 'psfonts.map' is read:

1. If a line is empty or begins with a space, percent, asterisk, semicolon, or hash mark, it is ignored.
2. Otherwise, the line is separated into words, where words are separated by spaces or tabs, except that if a word begins with a double quote, it extends until the next double quote or the end of the line.
3. If a word starts with '<<', it is taken as a font file to be wholly downloaded. Use this to avoid partial downloading, as described above.
4. If a word starts with '<[', it is taken as an encoding file to be downloaded. Use this if the name of the encoding file does end in '.enc', also as described above.
5. If a word starts with a '<' character, it is treated as a header file that needs to be downloaded. If the name ends in '.pfa' or '.pfb', it is taken as Type 1 font file that will be partially downloaded if the 'j' option is in effect. There can be more than one such header for a given font. If a '<' is a word by itself, the next word is taken as the name of the header file.
6. If a word starts with a '"' character, it is taken as PostScript code used in generating that font, and is inserted into the output verbatim at the appropriate point. (And 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 there is only one name, it is used for both the TeX name and the PostScript name.

7 Color

Dvips supports one-pass multi-color printing of TeX documents on any color PostScript device. Initially added by Jim Hafner, IBM Research, `hafner@almaden.ibm.com`, the color support has gone through many changes by Tomas Rokicki. Besides the source code support itself, there are additional TeX macro files: '`colordvi.tex`' and '`blackdvi.tex`', and corresponding '`.sty`' versions for use with LaTeX.

In this section we describe the use of color from the document preparer's point of view and then add some instructions on installation for the TeX administrator.

7.1 Color macro files

All the color macro commands are defined in '`colordvi.tex`' (or '`colordvi.sty`'). To access these macros simply add to the top of your plain TeX file the command:

```
\input colordvi
```

For (the obsolete) LaTeX 2.09, add the '`colordvi`' style option as in:

```
\documentstyle[12pt,colordvi]{article}
```

For LaTeX 2e, these examples are not applicable. Instead, please see the documentation for the graphics package, available from '*CTAN*`:doc/latex/graphics/`'. See also '*CTAN*`:doc/epslatex.ps`'.

These macros provide two basic kinds of color macros: ones for local color changes (a few words, a single symbol) and one for global color changes (the whole document). All the color names use a mixed case scheme to avoid conflicts with other macros. There are 68 predefined colors, with names taken primarily from the Crayola crayon box of 64 colors, and one pair of macros for the user to set his own color pattern (see Section 7.2 [User-definable colors], page 48). You can browse the file '`colordvi.tex`' for a list of the predefined colors. The comments in this file also show a rough correspondence between the crayon names and Pantones.

A local color command has the form

`\`*ColorName*`{this is the color `*ColorName*`}`

where *ColorName* is the name of a predefined color, e.g., '`Blue`'. As shown, these macros take one argument, the text to print in the specified color. This can be used for nested color changes since it restores the original color state when it completes. For example:

```
This text is normal but here we are \Red{switching to red,
\Blue{nesting blue}, recovering the red} and back to original.
```

The color nesting level has no hard limit, but it is not advisable to nest too deeply lest you and the reader lose track of the color history.

The global color command has the form

`\text`*ColorName*

These macros take no arguments and changes the default color from that point on to *ColorName*. This of course can be overridden globally by another such command or locally by local color commands. For example, expanding on the example above, we might have

```
\textGreen
This text is green but here we are \Red{switching to red,
\Blue{nesting blue}, recovering the red} and back to
original green.
\textCyan
The text from here on will be cyan until
\Yellow{locally changed to yellow}. Now we are back to cyan.
```

The color commands will even work in math mode and across math mode boundaries. This means that if you have a color before going into math mode, the mathematics will be set in that color as well. In alignment environments like `\halign`, '`tabular`' or '`eqnarray`', local color commands cannot extend beyond the alignment characters.

Because local color commands respect only some environment and delimiter changes besides their own, care must be taken in setting their scope. It is best not to have them stretch too far.

At the present time there are no macros for color environments in LaTeX which might have a larger range. This is primarily to keep the TeX and LaTeX use compatible.

7.2 User-definable colors

There are two ways for the user to specify colors not already defined. For local changes, there is the command `\Color` which takes two arguments. The first argument is four numbers between zero and one and specifies the intensity of cyan, magenta, yellow and black (CMYK) in that order. The second argument is the text that should appear in the given color. For example, suppose you want the words "this color is pretty" to appear in a color which is 50% cyan, 85% magenta, 40% yellow and 20% black. You would use the command

```
\Color{.5 .85 .4 .2}{this color is pretty}
```

For global color changes, there is a command `\textColor` which takes one argument, the CMYK quadruple of relative color intensities. For example, if you want the default color to be as above, then the command

```
\textColor{.5 .85 .4 .2}
The text from now on will be this pretty color
```

will do the trick.

Making a global color change in the midst of nested local colors is highly discouraged. Consequently, Dvips will give you warning message and do its best to recover by discarding the current color history.

7.3 Color subtleties

Color macros are defined via `\special` keywords. As such, they are put in the '`.dvi`' file only as explicit message strings to the driver. The (unpleasant) result is that certain unprotected regions of the text can have unwanted color side effects. For example, if a color region is split by TeX across a page boundary, then the footers of the current page (e.g., the page number) and the headers of the next page can inherit that color. To avoid this effect globally, users should make sure that these special regions of the text are defined with their own local color commands. For example, to protect the header and footer in plain TeX, use

```
\headline{\Black{My Header}}
\footline{\Black{\hss\tenrm\folio\hss}}
```

This warning also applies to figures and other insertions, so be careful!

Of course, in LaTeX, this is much more difficult to do because of the complexity of the macros that control these regions. This is unfortunate but inevitable, because TeX and LaTeX were not written with color in mind.

Even when writing your own macros, much care must be taken. The macros that 'colorize' a portion of the text work prefix the text work by outputting one `\special` command to turn the color on before the text, and outputting another `\special` command afterwards to restore the original color. It is often useful to ensure that TeX is in horizontal mode before the first special command is issued; this can be done by prefixing the color command with `\leavevmode`.

7.4 Printing in black/white after colorizing

If you have a TeX or LaTeX document written with color macros and you want to print it in black and white there are two options. On all (good) PostScript devices, printing a color file will print in corresponding gray levels. This is useful to get a rough idea of the colors without using expensive color printing devices. The second option is to replace the call to input '`colordvi.tex`' with '`blackdvi.tex`' (and similarly for the '`.sty`' files). So in the above example, replacing the word '`colordvi`' with '`blackdvi`' suffices. '`blackdvi.tex`' defines the color macros as no-ops, and so will produce normal black/white printing. By this simple mechanism, the user can switch to all black/white printing without having to ferret out the color commands. Also, some device drivers, particularly non-PostScript ones like screen previewers, will simply ignore the color commands and so print in black/white. Hopefully, in the future screen previewers for color displays will be compatible with some form of color support.

7.5 Color device configuration

To configure Dvips for a particular color device you need to fine tune the color parameters to match your device's color rendition. To do this, you will need a Pantone chart for your device. The header file '`color.lpro`' shows a (rough) correspondence between the Crayola crayon names and the Pantone numbers and also defines default CMYK values for each of the colors. Note that these colors must be defined in CMYK terms and not RGB, as Dvips outputs PostScript color commands in CMYK. This header file also defines (if they are not known to the interpreter) the PostScript commands '`setcmykcolor`' and '`currentcmykcolor`' in terms of a RGB equivalent so if your device only understands RGB, there should be no problem.

The parameters set in this file were determined by comparing the Pantone chart of a Tektronix Phaser printer with the actual Crayola Crayons. Because these were defined for a particular device, the actual color rendition on your device may be very different. There are two ways to adjust this. One is to use the PAntone chart for your device to rewrite '`color.lpro`' prior to compilation and installation. A better alternative, which supports multiple devices, is to add a header file option in the configuration file (see Section 3.4.2

[Configuration file commands], page 16) for each device that defines, in 'userdict', the color parameters for those colors that need redefining.

For example, if you need to change the parameters defining 'Goldenrod' (approximately Pantone 109 on the Phaser) for your device 'mycolordev', do the following. In the Pantone chart for your device, find the CMYK values for Pantone 109. Let's say they are '{\ 0 0.10 0.75 0.03 }'. Then create a header file named 'mycolordev.pro' with the commands

```
userdict begin
/Goldenrod { 0 0.10 0.75 0.03 setcmykcolor} bind def
```

Finally, in 'config.mycolordev' add the line

```
h mycolordev.pro
```

This will then define 'Goldenrod' in your device's CMYK values in 'userdict' which is checked before defining it in 'TeXdict' by 'color.pro'. (On MS-DOS, you will have to call this file 'mycolordev.cfg'.)

This mechanism, together with additions to 'colordvi.tex' and 'blackdvi.tex' (and the '.sty' files), can also be used to predefine other colors for your users.

7.6 Color support details

To support color, Dvips recognizes a certain set of specials. These specials start with the keyword 'color' or the keyword 'background', followed by a color specification.

7.6.1 Color specifications

What is a color specification? One of three things. First, it might be a PostScript procedure as defined in a PostScript header file. The 'color.pro' file defines 64 of these, including 'Maroon'. This PostScript procedure must set the current color to be some value; in this case, 'Maroon' is defined as '0 0.87 0.68 0.32 setcmykcolor'.

The second possibility is the name of a color model (initially, one of 'rgb', 'hsb', 'cmyk', or 'gray') followed by the appropriate number of parameters. When Dvips encounters such a macro, it sends out the parameters first, followed by the string created by prefixing 'TeXcolor' to the color model. Thus, the color specification 'rgb 0.3 0.4 0.5' would generate the PostScript code '0.3 0.4 0.5 TeXrgbcolor'. Note that the case of zero arguments is disallowed, as that is handled by the single keyword case ('Maroon') above, where no changes to the name are made before it is sent to the PostScript file.

The third and final type of color specification is a double quote followed by any sequence of PostScript. The double quote is stripped from the output. For instance, the color specification '"AggiePattern setpattern' will set the 'color' to the Aggie logo pattern (assuming such exists.)

7.6.2 Color specials

We will describe 'background' first, since it is the simplest. The 'background' keyword must be followed by a color specification. That color specification is used as a fill color for the background. The last 'background' special on a page is the one that gets issued, and it gets issued at the very beginning of the page, before any text or specials are sent. (This is possible because the prescan phase of Dvips notices all of the color specials so that the appropriate information can be written out during the second phase.)

The '`color`' special itself has three forms. The first is just '`color`' followed by a color specification. In this case, the current global color is set to that color; the color stack must be empty when such a command is executed.

The second form is '`color push`' followed by a color specification. This saves the current color on the color stack and sets the color to be that given by the color specification. This is the most common way to set a color.

The final version of the '`color`' special is just '`color pop`', with no color specification; this says to pop the color last pushed on the color stack from the color stack and set the current color to be that color.

Dvips correctly handles these color specials across pages, even when the pages are repeated or reversed.

These color specials can be used for things such as patterns or screens as well as simple colors. However, note that in the PostScript, only one color specification can be active at a time. For instance, at the beginning of a page, only the bottommost entry on the color stack is sent; also, when a color is popped, all that is done is that the color specification from the previous stack entry is sent. No '`gsave`' or '`grestore`' is used. This means that you cannot easily mix usage of the '`color`' specials for screens and colors, just one or the other. This may be addressed in the future by adding support for different categories of color-like state.

Index

!

%

(

-

.

B

C

D

H

I

J

K

L

M

T

U

V

W

X

Y

Z

Short Contents

Table of Contents