## 16.5 Configuring dvips for Color Devices

To configure dvips for a particular color device you need to fine tune the color parameters to match your devices 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 Tektronics 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 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` (approximate PANTONE 109) 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 } 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`.

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

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 a 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.

## 16.3 Subtleties in Using Color

These color macros are defined by use of specialized `\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 in TEX, to protect the header and footer, use

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

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 is somehow inevitable because TEX and LaTEX were not written with color in mind.

## 16.4 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 grey-levels. This is useful since in this way you can get a rough idea of where the colors are changing without using expensive color printing devices. The second option is to replace the call to input `dpscolor.tex` with `dpsblack.tex` (and similarly for the `.sty` files). So in the above example, replacing the word `dpscolor` with `dpsblack` suffices. This file 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 normal black/white. Hopefully, in the future screen previewers for color displays will be compatible with some form of color support.

In principle there is no limit to the nesting level, but it is not advisable to nest too deep lest you loose track of the color history.

The global color command has the form

```
\textColorName
```

This macro takes no arguments and immediately changes the default color from that point on to the specified color. This of course can be overriden 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 unless
\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. More importantly however, 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 deliminator changes besides their own, care must be taken in setting their scope. It is best not to have then 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.

## 16.2 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 a quadruple of 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}
```

# 16. Using Color with dvips

This new feature of `dvips` is somewhat experimental so your experiences and comments are welcome. It was added by Jim Hafner, IBM Research, `hafner@almaden.ibm.com`, so please direct your comments there. Besides the changes to the code itself, there are additional TEX macro files `dpscolor.tex` and `dpsblack.tex`. There are also `.sty` versions of these files that can be used with LaTEX and other similar macro packages. This feature adds one-pass multi-color printing of TEX documents on any color PostScript device.

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

## 16.1 The Macro Files

All the color macro commands are defined in `dpscolor.tex` (or `dpscolor.sty`). To access these macros simply add to the top of your TEX file the command

```
\input dpscolor
```

or, if your document uses style files like LaTEX, add the `dpscolor` style option as in

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

There are basically two kinds of color macros, ones for local color changes to, say, a few words or even one symbol and one for global color changes. Note that all the color names use a mixed case scheme. 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. More on this extra feature later. You can browse the file `dpscolor.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 is in the form

```
\ColorName{this will print in color}
```

Here `ColorName` is the name of a predefined color. As this example shows, this type of command takes one argument which is the text that is to print in the selected color. This can be used for nested color changes since it restores the original color state when it completes. For example, suppose you were writing in green and want to switch temporarily to red, then blue, back to red and restore green. Here is one way that you can do this:

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

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 hex) will probably fail with `dvips`.

## 15.8 Can't Find Font Files

If `dvips` complains that it cannot find certain font files, it is possible that the paths haven't been set up correctly for your system. Use the debug flags to determine precisely what fonts are being looked for and make sure these match where the fonts are located on your system.

## 15.9 Can't Generate Fonts

This happens a lot if either `MakeTeXPK` hasn't been properly edited and installed, or if the local installation of METAFONT isn't correct. If `MakeTeXPK` isn't properly edited or isn't installed, an error such as `MakeTeXPK not found` will be printed on the console. The fix is to talk to the person who installed `dvips` and have them fix this.

If METAFONT isn't found when `MakeTeXPK` is running, make sure it is installed on your system. The person who installed TEX should be able to install METAFONT easily.

If METAFONT runs but generates fonts that are too large (and prints out the name of each character as well as just a character number), then your METAFONT base file probably hasn't been made properly. To make a proper `plain.base`, assuming the local mode definitions are contained in `local.mf` (on the NeXT, `next.mf`; on the Amiga, `amiga.mf`), type the following command (assuming `csh` under UNIX):

```
localhost> inimf "plain; input local; dump"
```

Now, copy the `plain.base` file from the current directory to where the base files are stored on your system.

Note that a preloaded `cmbase.base` should never be used when creating fonts, and a program such as `cmmf` should never exist on the system. The macros defined in `cmbase` will break fonts that do not use `cmbase`; such fonts include the LaTEX fonts. Loading the `cmbase` macros when they are needed is done automatically and takes less than a second—an insignificant fraction of the total run time of METAFONT for a font, especially when the possibility of generating incorrect fonts is taken into account. If you create the LaTEX font `circle10`, for instance, with the `cmbase` macros loaded, the characters will have incorrect widths.

## 15.4 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 make sure you 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 telling `dvips` that the printer has more memory than it actually does, and then printing a complicated document. To fix this, try lowering the parameter to `m` in the configuration file; use the debug option to make sure you adjust the correct file.

Other errors may indicate that the graphics you are trying to include don't nest properly in other PostScript documents, or any of a number of other possibilities. Try the output on a QMS PS-810 or other Adobe PostScript printer; it might be a problem with the printer itself.

## 15.5 400 DPI Is Used Instead Of 300 DPI

This common error is caused by not editing the `config.ps` file to reflect the correct resolution for your site. You can use the debug flags (`-d64`) to see what files are actually being read.

## 15.6 Long Documents Fail To Print

This is usually caused by incorrectly specifying the amount of memory the printer has in `config.ps`; see the description above.

## 15.7 Including Graphics Fails

The reasons why graphics inclusions fail are too numerous to mention. The most common problem is an incorrect bounding box; read the section on bounding boxes and check your PostScript file. Complain very loudly 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 should 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, to strip the comments from an included figure. This might be necessary if the PostScript spooling software does not read the structuring

## 15.1 Debug Options

The `-d` flag to `dvips` is very useful for helping to track 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. The classes are:

| | |
|---|---|
| 1 | specials |
| 2 | paths |
| 4 | fonts |
| 8 | pages |
| 16 | headers |
| 32 | font compression |
| 64 | files |
| 128 | memory |

## 15.2 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. I've tried to work around as many bugs as possible in common non-Adobe PostScript interpreters, but I'm sure I've missed a few.

If `dvips` gives any strange error messages, or compilation on your machine generated a lot of warnings, perhaps the `dvips` program itself is broken. Carefully check the types in `structures.h` and the declarations in the `Makefile`, and try using the debug options to determine where the error occurred.

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.

## 15.3 Output Too Small or Inverted

If some documents come out inverted or too small, your spooler is not supplying an end of job indicator at the end of each file. (This happens a lot on small machines that don't have spoolers.) You can force `dvips` to do this with the `-F` flag, but note that this generates files with a binary character (control-D) in them. You can also try using the `-s` flag to enclose the entire job in a save/restore pair.

Now, check the configuration parameters in `config.ps`. You should also update the default resolution here. This file is the system-wide configuration file that will be automatically installed. If you are unsure how much memory your PostScript printer has, print the following file:

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

Note that the number returned by this file is the total memory free; it is often a good idea to tell `dvips` that the printer has somewhat less memory. This is because many programs download permanent macros that can reduce the memory in the printer. In general, a memory size of about `300000` is plenty, since `dvips` can automatically split a document if required. It is unfortunate that PostScript printers with much less virtual memory still exist. Some systems or printers can dynamically increase the memory available to a PostScript interpreter; for these systems, a value of one million works well.

Next, run `make`. Everything should compile smoothly. You may need to adjust the compiler options in the `Makefile` if something goes amiss.

Once everything is compiled, run `make install`. After this is done, you may want to create a configuration file for each PostScript printer at your site.

If the font caching is considered a security hole, make the 'cache' directory be something like `/tmp/pks`, and `cron` a job to move the good `pk` files into the real directory. Or simply disable this feature by not installing `MakeTeXPK`.

Don't forget to install the new `vf` files and `tfm` files. Note that the `tfm` files distributed with earlier (pre-5.471) versions of `dvips`, and all versions of other PostScript drivers, are different.

# 15. 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. This is all too common, and is usually caused by some broken PostScript application out there. The following sections provide some helpful hints if you find yourself in such a situation.

In all cases, you should attempt to find the smallest file that causes the problem. This will not only make debugging easier, it will also reduce the number of possible interactions among different parts of the system.

# 13. Other Bells And Whistles

For special effects, if any of the macros `bop-hook`, `eop-hook`, `start-hook`, or `end-hook` are defined in the PostScript `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.

As an example of what can be done, the following special will write a light 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}
```

Note that using `bop-hook` or `eop-hook` in any way that preserves information across pages will break compliance with the Adobe document structuring conventions, so if you use any such tricks, it is recommended that you also use the `-N` option to turn off structured comments.

Several of the above tricks can be used nicely together, and it is not necessary that a 'printer configuration file' be used only to set printer defaults. For instance, a `-P` file can be set up to print the date on each page; the particular configuration file will execute a command to put the date into a header file, which is then included with a `h` line in the configuration file. Note that multiple `-P` options can be used.

# 14. Installation

If `dvips` has not already been installed on your system, the following steps are all that is needed.

First update the `Makefile`—in particular, the paths. Everything concerning `dvips` can be adjusted in the `Makefile`. Make sure you set key parameters such as the default resolution, and make sure that the path given for packed pixel files is correct.

Next, check the file name definitions in `MakeTeXPK`. If you don't have METAFONT installed, you cannot use `MakeTeXPK` to automatically generate the fonts; you can, however, modify it to generate `pk` fonts from `gf` fonts if you don't have a full set of `pk` fonts but do have a set of `gf` fonts. If you don't have that, you should probably not install `MakeTeXPK` at all; this will disable automatic font generation.

# 12. Environment Variables

The `dvips` program reads a certain set of environment variables to configure its operation. The path variables are read after all configuration files are read, so they override values in the configuration files. (The `TEXCONFIG` variable, of course, is read before the configuration files.) The rest are read as needed.

`HOME` (no default) This environment variable is automatically set by the shell and is used to replace any occurrences of ~ in a path.

`MAKETEXPK` (`MakeTeXPK %n %d %b %m`) This environment variable sets the command to be executed to create a missing font. A %n is replaced by the base name of the font to be created (such as `cmr10`); a %d is replaced by the resultant horizontal resolution of the font; a %b is replaced by the horizontal resolution at which `dvips` is currently generating output, and any %m is replaced by a string that METAFONT can use as the right hand side of an assignment to `mag` to create the desired font at the proper resolution. If a mode for METAFONT is set in a configuration file, that is automatically appended to the command before execution. Note that these substitutions are different than the ones performed on PK paths.

`PRINTER` (no default) This environment variable is read to determine which default printer configuration file to read in. Note that it is the responsibility of the configuration file to send output to the proper print queue, if such functionality is desired.

`TEXFONTS` (`/LocalLibrary/Fonts/TeXFonts/tfm:/usr/lib/tex/fonts/tfm`) This is where `tfm` files are searched for. A `tfm` file only needs to be loaded if the font is a resident (PostScript) font or if for some reason no `pk` file could be found.

`TEXPKS` (`/LocalLibrary/Fonts/TeXFonts/pk:/usr/lib/tex/fonts/pk`) This environment variable is a path on which to search for `pk` fonts. Certain substitutions are performed if a percent sign is found anywhere in the path. A `%d` is replaced by the resolution at which the font is desired; a `%f` is replaced by the base file name of the font (such as `cmr10`); a `%m` is replaced by the METAFONT mode for this font; a `%p` is replaced by `pk`; and finally, `%%` is replaced by a single literal `%`.

`TEXINPUTS` (`.:..:/usr/lib/tex/inputs`) This environment variable is used to find PostScript figures when they are included.

`TEXCONFIG` (`.:/usr/lib/tex/ps`) This environment variable sets the directories to search for configuration files, including the system-wide one. Using this single environment variable and the appropriate configuration files, it is possible to set up the program for any environment. (The other path environment variables can thus be redundant.)

`VFFONTS` (`.:/usr/lib/tex/fonts/vf`) This environment variable sets where `dvips` looks for virtual fonts. A correct virtual font path is essential if PostScript fonts are to be used.

`MakeTeXPK` can be customized to do other things to get the font. For instance, if you are installing `dvips` to replace (or run alongside) an existing PostScript driver, and that driver demands `gf` fonts, you can easily modify `MakeTeXPK` to invoke `gftopk` to convert the `gf` files to `pk` files for `dvips`. This provides the same space savings listed above.

Because `dvips` (and thus `MakeTeXPK`) is run by a wide variety of users, there must be a system-wide place to put the cached font files. In order for everyone to be able to supply fonts, the directory must be world writable. If your system administrator considers this a security hole, `MakeTeXPK` can write to `/tmp/pk` or some such directory, and periodically the cached fonts can be moved to a more general system area. Note that the cache directory must exist on the `pk` file search path in order for `MakeTeXPK` to work.

# 11. Path Interpretation

The `dvips` program needs to read a wide variety of files from a large set of directories. It uses a set of paths to do this. The actual paths are listed in the next section; this section describes how the paths are interpreted.

All path variables are names of directories (path elements), separated by colons. Each path element can be either the literal name of a directory or one of the `~` forms common under UNIX. If a path element is a single tilde, it is replaced by the contents of the environment variable `HOME`, which is normally set to the user's home directory. If the path element is a tilde followed by anything, the part after the tilde is interpreted as a user name, and his home directory is fetched from the system password file and used as the real path element.

Where environment variables can override paths, an additional path element form is allowed. If a path element is the empty string, it is replaced with the system defaults. Thus, to search the user's home directory, followed by the system default paths, assuming the current shell is `csh`, the following command would be used:

```
setenv TEXINPUTS ~:
```

This is a path of two elements. The first is the user's home directory. The second path element is the empty string, indicating that the system defaults should be searched.

The 'system defaults' as defined here means the strings set in the `Makefile` before compilation, rather than any defaults set in `config.ps` or printer-specific configuration files. This is to prevent path blowup, where more and more directories are added to the path by each level of configuration file.

`S` *path*: The path to search for special illustrations (Encapsulated PostScript files or psfiles) is *path*. The `TEXINPUTS` environment variable will override this.

`T` *path*: The path to search for the `tfm` files is *path*. The `TEXFONTS` environment variable will override this. This path is used for resident fonts and fonts that can't otherwise be found. It's usually best to make it identical to the path used by TEX.

`U`: Turns off a memory-saving optimization; this is necessary for the Xerox 4045 printer, but not recommended otherwise. See the description above for more information.

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

`W` *string*: Sends *string* to stderr, if a parameter is given; otherwise it cancels another previous message. This is useful in the default configuration file if you want to require the user to specify a printer, for instance, or if you want to notify the user that the resultant output has special characteristics.

`X` *num*: Set the horizontal resolution to *num* dots per inch.

`Y` *num*: Set the vertical resolution to *num* dots per inch.

`Z`: Compress all downloaded fonts by default, as above.

# 10. Automatic Font Generation

One major problem with TEX and the Computer Modern fonts is the huge amount of disk space a full set of high resolution fonts can take. The `dvips` program solves this problem by creating fonts on demand, so only those fonts that are actually used are stored on disk. At a typical site, less than one-fifth of the full set of Computer Modern fonts are used over a long period, so this saves a great deal of disk space.

Furthermore, the addition of dynamic font generation allows fonts to be used at any size, including typesetter resolutions and extremely huge banner sizes. Nothing special needs to be done; the fonts will be automatically created and installed as needed.

The downside is that it does take a certain amount of time to create a new font if it has never been used before. But once a font is created, it will exist on disk, and the next time that document is printed it will print very quickly.

It is the `MakeTeXPK` shell script that is responsible for making these fonts. The `MakeTeXPK` script supplied invokes METAFONT to create the font and then copies the resultant `pk` file to a world-writable font cache area.

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.

`q`: Run in quiet mode by default.

`r`: Reverse the order of pages by default.

`s`: Enclose the entire document in a global save/restore pair by default. Not recommended, but useful in some environments; this breaks the conformance of the document to the Adobe PostScript structuring conventions.

`D` *num*: Set the vertical and horizontal resolution to *num* dots per inch. Useful in printer-specific configuration files.

`E` *command*: Executes the command listed; can be used to get the current date into a header file for inclusion, for instance. Possibly dangerous; in many installations this may be disabled, in which case a warning message will be printed if the option is used.

`H` *path*: The (colon-separated) path to search for PostScript header files is *path*.

`K`: Filter comments out of included PostScript files; see the description above for more information.

`M` *mode*: Set *mode* as the METAFONT mode to be used when generating fonts. This is passed along to `MakeTeXPK` and overrides mode derivation from the base resolution.

`N`: Disable PostScript comments by default.

`O` *offset*: Move the origin by a certain amount. The *offset* is a comma-separated pair of dimensions, such as `.1in,-.3cm` (in the same syntax as used in the `papersize` special). 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.

`P` *path*: The (colon-separated) path to search for bitmap `pk` font files is *path*. The `TEXPKS` environment variable will override this. If a `%` character is found in *path*, the following substitutions will be made, and then a search will be made for the resulting filenames. A `%f` is replaced by the font name. A `%d` is replaced by the font size in dots per inch. A `%p` is replaced by the font family; this is always `pk`. A `%m` is replaced by the font mode; this is the mode given in the `M` option.

`R` *num num* ...: Sets up a list of default resolutions to search for `pk` fonts, if the requested size is not available. The output will then scale the font found using PostScript scaling to the requested size. Note that the resultant output will be ugly, and thus a warning is issued. To turn this off, use a line with just the `R` and no numbers.

should not send any PostScript commands down (although a structured comment or two would be okay). See the supplied `config.ps` file for a more realistic example.

`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.

`e` *num*: Set the maximum drift parameter to *num* dots (pixels) as explained above.

`f`: Run as a filter by default.

`h` *name*: Add *name* as a PostScript header file to be downloaded at the beginning.

`i` *num*: Make each section be a separate file, and set the maximum number of pages in a given file to *num*. Under certain circumstances, `dvips` will split the document into 'sections' to be processed independently; this is most often done for memory reasons. Using this option tells `dvips` to place each section into a separate file; the new file names are created by adding a two-digit sequence number to the end of the supplied file name. This is essentially a combination of the command line options `-i` and `-S`; see the documentation for these options for more information.

`m` *num*: The value *num* is the virtual 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
```

Note that the number returned by this file is the total memory free; it is often a good idea to tell `dvips` that the printer has somewhat less memory. This is because many programs download permanent macros that can reduce the memory in the printer. In general, a memory size of about `300000` is plenty, since `dvips` can automatically split a document if required. It is unfortunate that PostScript printers with much less virtual memory still exist. 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; the NeXT computer is such a machine. For these systems, a value of one million works well.

`o` *name*: The default output file is set to *name*. As above, it can be a pipe. Useful in printer-specific configuration files to redirect the output to a particular printer queue.

`p` *name*: The file to examine for PostScript font aliases is *name*. It defaults to `psfonts.map`. This option allows different printers to use different resident fonts. If the name starts with a '`+`' character, then the rest of the name (after any leading spaces) is used as

Within a configuration file, any empty line or line starting with a space, asterisk, equal sign, or a pound sign is ignored.

`@` *name hsize vsize*: This option is used to set the paper size defaults and options for the particular printer this configuration file describes. There are three formats for this option. If the option is specified on a line by itself, with no parameters, it instructs `dvips` to discard all other paper size information (possibly from another configuration file) and start fresh. If three parameters are given, as above, 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* describe the horizontal and vertical size of the sheet of paper, respectively. If both *hsize* and *vsize* are equal to zero (although you must still specify units!) then any page size will match it. 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 to select that particular paper size. 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 instance, a subset of the paper size information supplied in the default `config.ps` looks like

```
@ letterSize 8.5in 11in

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

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

Note that you can even include structured comments in the configuration file! 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` comment, 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 given is used, so the first paper size should always be the default. The dimensions must match within a quarter of an inch. Landscape mode for all of the paper sizes are 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. Note that 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

# 8. Configuration File Searching

The dvips program has a system of loading configuration files such that certain parameters can be set globally across the system, others can be set on a per-printer basis, and yet others can be set by the user. When dvips starts up, first the global config.ps file is searched for and loaded. This file is looked for along the path for header files, which is by default .:/usr/lib/tex/ps. After this master configuration file is loaded, a file by the name of .dvipsrc is loaded from the current user's home directory, if such a file exists. This file is loaded in exactly the same way as the global configuration file, and it can override any options set in the global file.

Then the command line is read and parsed. If the -P option is encountered, at that point in the command line a configuration file for that printer is read in. Thus, the printer configuration file can override anything in the global or user configuration file, and it can also override anything seen in the command line up to the point that the -P option was encountered.

After the command line has been completely scanned, if there was no -P option selected, and also the -o and -f command line options were not used, a PRINTER environment variable is searched for. If this variable exists, and a configuration file for the printer mentioned in it exists, this configuration file is loaded last of all.

Note that because the printer-specific configuration files are read after the user's configuration file, the user cannot override things in the printer configuration files. On the other hand, the header path usually includes the current directory, and can be set to include the user's home directory (or any other directory of the user), so the user can always provide his or her own printer-specific configuration files that will be found before the system global ones.

If your printer uses a different resolution than 300 dpi, make sure that you have given a METAFONT mode as well as a resolution in the printer configuration file. Also make sure that METAFONT knows about the mode, by entering it into your local mode_def file (typically waits.mf; amiga.mf on the Amiga, next.mf on the NeXT) and recreating the plain.base file for METAFONT, including the mode_def file. (Another good mode definition file is modes.mf by Karl Berry, which is available from ftp.cs.umb.edu in pub/tex/modes.mf.) The most common problem in generating fonts with METAFONT is that this file with the mode definitions is not included when creating the plain.base file.

# 9. Configuration File Options

Most of the configuration file options are similar to the command line options, but there are a few new ones.

Again, many may be turned off by suffixing the letter with a zero (0). These options are a, f, q, r, K, N, U, and Z.

`-O` *offset*: Move the origin by a certain amount. The *offset* is a comma-separated pair of dimensions, such as `.1in,-.3cm` (in the same syntax used in the `papersize` special). 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.

`-P` *printername*: Sets up the output for the appropriate printer. This is implemented by reading in `config.`*printername*, which can then set the output pipe (as in, `o !lpr -Pprintername`) as well as the font paths and any other defaults for that printer only. It is recommended that all standard defaults go in the one master `config.ps` file and only things that vary printer to printer go in the `config.`*printername* files. Note that `config.ps` is read before `config.`*printername*. In addition, another file called `~/.dvipsrc` is searched for immediately after `config.ps`; this file is intended for user defaults. If no `-P` command is given, the environment variable `PRINTER` is checked. If that variable exists, and a corresponding configuration file exists, that configuration file is read in.

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

`-T` *printername*: Same as `-P`. However, all parameters are set to the defaults for *printername* before processing any other options, so the position of `-T` on the command line doesn't matter.

`-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. It is caused by a bug in that interpreter that results in 'garbage' on the bottom of each character. Not recommended unless you must drive this printer.

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

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

`-Z`: Causes bitmapped fonts to be compressed before they are downloaded, thereby reducing the size of the PostScript font-downloading information. Especially useful at high resolutions or when very large fonts are used. Will slow down printing somewhat, especially on early 68000-based PostScript printers.

`-C` *num*: Create *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 resolution in dpi (dots per inch) to *num.* 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.

`-E`: Makes `dvips` attempt to generate an EPSF file with a tight bounding box. This only works on one-page files, and it 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 lie 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 for creating small EPSF files for equations or tables or the like. (Note, of course, that `dvips` output 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 a great deal of care.)

`-F`: Causes Control-D (ASCII code 4) to be appended 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 extremely small systems. Otherwise, it is not recommended.

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

`-K`: This option causes comments in included PostScript graphics, font files, and headers to be removed. This is sometimes 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 some included graphics to fail, since the PostScript header macros from some software packages read portions of the input stream line by line, searching for a particular comment. Due to the universality of buggy PostScript previewers and spoolers, this is enabled by default; use `-K0` to turn it off.

`-M`: Turns off the automatic font generation facility. If any fonts are missing, commands to generate the fonts are appended to the file `missfont.log` in the current directory; this file can then be executed and deleted to create the missing fonts.

`-L`: Set landscape mode.

`-N`: Turns off structured comments; this might 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.

`-n` *num*: At most *num* pages will be printed. Default is 100000.

`-o` *name*: The output will be sent to file *name*. If no file name is given, the default name is *file*`.ps` where the `dvi` file was called *file*`.dvi`; if this option isn't given, the default name is `!lpr`. If the first character of the file name is an exclamation mark, 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. This option also disables the automatic reading of the `PRINTER` environment variable, and turns off the automatic sending of control D if it was turned on with the `-F` option or in the configuration file; use `-F` after this option if you want both.

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

`-q`: Run in quiet mode. Don't chatter about pages converted, etc.; report nothing but errors to standard error.

`-r`: Stack pages in reverse order. Normally, page 1 will be printed first.

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

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

`-x` *num*: Set the magnification ratio to *num*/1000. Overrides the magnification specified in the `dvi` file. Must be between 10 and 100000.

`-A` *num*: Put *num* pages of the `dvi` file onto one output page.

`-B` *num*: Assume that the printer has *num* bytes of memory (see the `m` parameter in the Configuration File Options section, below).

`-d` *num*: Set the debug flags. This is intended only for emergencies or for unusual fact-finding expeditions; it will work only if `dvips` has been compiled with the `DEBUG` option. The source file `debug.h` indicates what the values of *num* can be, or see section 15 of this manual. Use a value of $-1$ for maximum output.

`-e` *num*: Make sure that each character is placed at most this many pixels from its 'true' resolution-independent position on the page. The default value of this parameter is resolution dependent (it is the number of entries in the list [100, 200, 300, 400, 500, 600, 800, 1000, 1200, 1600, 2000, 2400, 2800, 3200, ...] that are less than or equal to the resolution in dots per inch). Allowing individual characters to 'drift' from their correctly rounded positions by a few pixels, while regaining the true position at the beginning of each new word, improves the spacing of letters in words.

`-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 you must use 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, and turns off the automatic sending of control D if it was turned on with the `-F` option or in the configuration file; use `-F` after this option if you want both.

`-h` *name*: Prepend file *name* as an additional header file. (However, if the name is simply '`-`', suppress all header files from the output.) This header file gets added to the PostScript `userdict`.

`-i`: Make each section be a separate file. Under certain circumstances, `dvips` will split the document up into 'sections' to be processed independently; this is most often done for memory reasons. Using this option tells `dvips` to place each section into a separate file; the new file names are created by adding a two-digit sequence number to the end of the supplied file name. This option is most often used in conjunction with the `-S` option which sets the maximum section length in pages. 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.

`-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.

`-l` *num*: The last page printed will be the first one numbered *num*. Default is the last page in the document. If the *num* is prefixed by an equals sign, then it (and any argument to the `-p` option) is treated as a sequence number, rather than a value to compare with `\count0` values. 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 for printer.

Please contact Karl Berry if you have any comments or additions. Karl can be reached at `karl@cs.umb.edu`, or at 135 Center Hill Road, Plymouth, MA 02360.

# 7. Command Line Options

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

Many of the parameterless options listed here can be turned off by immediately suffixing the option with a zero (0); for instance, to turn off page reversal if it is turned on by default, use `-r0`. The options that can be turned off in this way are `a`, `f`, `k`, `i`, `m`, `q`, `r`, `s`, `E`, `F`, `K`, `M`, `N`, `U`, and `Z`.

This is a handy summary of the options; it is printed out when you run `dvips` with no arguments.

```
    Usage: dvips [options] filename[.dvi]
 a*  Conserve memory, not time      t s Paper format
 c # Uncollated copies              x # Override dvi magnification
 d # Debugging                      C # Collated copies
 e # Maxdrift value                 D # Resolution
 f*  Run as filter                  E*  Try to create EPSF
 h f Add header file                F*  Send control-D at end
 i*  Separate file per section      K*  Pull comments from inclusions
 k*  Print crop marks               M*  Don't make fonts
 l # Last page                      N*  No structured comments
 m*  Manual feed                    O c Set/change paper offset
 n # Maximum number of pages        P s Load config.$s
 o f Output file                    S # Max section size in pages
 p # First page                     T c Specify desired page size
 q*  Run quietly                    U*  Disable string param trick
 r*  Reverse order of pages         X # Horizontal resolution
 s*  Enclose output in save/restore Y # Vertical resolution
                                    Z*  Compress bitmap fonts
    # = number   f = file   s = string  * = suffix, `0' to turn off
    c = comma-separated dimension pair (e.g., 3.2in,-32.1cm)
```

`-c` *num*: Generate *num* copies of every page. Default is 1. (For collated copies, see the `-C` option below.)

`-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.

## 6.7 Examples

In closing, I will give two examples. First, the fonts in the Univers typeface family were assigned numbers by its designer, Adrien Frutiger. (You can see the scheme on, for example, page 29 of *The Art of Typo.icon.ography*, by Martin Solomon.) Naturally, we want to give them names.

| | |
|---|---|
| `unl` | 45 (light) |
| `unli` | 46 (light italic) |
| `unlrc` | 47 (light condensed) |
| `unlic` | 48 (light condensed italic) |
| `unlro` | 49 (light extra condensed) |
| `unmrx` | 53 (medium extended) |
| `unm` | 55 (medium) |
| `unmi` | 56 (medium italic) |
| `unmrc` | 57 (medium condensed) |
| `unmic` | 58 (medium condensed italic) |
| `unmro` | 59 (medium extra condensed) |
| `undrx` | 63 (demibold extended) |
| `und` | 65 (demibold) |
| `undi` | 66 (demibold italic) |
| `undrc` | 67 (demibold condensed) |
| `undic` | 68 (demibold condensed italic) |
| `unbrx` | 73 (bold extended) |
| `unb` | 75 (bold) |
| `unbi` | 76 (bold italic) |
| `unxrx` | 83 (extra bold extended) |

Second, here are names for the standard PostScript fonts and their variants: Fonts marked by an asterisk do not require using virtual fonts; the raw fonts can be used directly because no remapping is necessary; every character is encoded.

| | |
|---|---|
| `pagk` | AvantGarde-Book |
| `pagkc` | AvantGarde-Book (Small Caps) |
| `pagko` | AvantGarde-BookOblique |
| `pagd` | AvantGarde-Demi |
| `pagdo` | AvantGarde-DemiOblique |
| `pbkd` | Bookman-Demi |
| `pbkdi` | Bookman-DemiItalic |
| `pbkl` | Bookman-Light |
| `pbkli` | Bookman-LightItalic |
| `pbklc` | Bookman-Light (Small Caps) |
| `pcrb` | Courier-Bold |
| `pcrbo` | Courier-BoldOblique |
| `pcrro` | Courier-Oblique |
| `pcrr` | Courier |
| `phvb` | Helvetica-Bold |
| `phvbo` | Helvetica-BoldOblique |
| `phvro` | Helvetica-Oblique |
| `phvr` | Helvetica |
| `phvrc` | Helvetica (Small Caps) |
| `phvbrn` | Helvetica-Narrow-Bold |
| `phvbon` | Helvetica-Narrow-BoldOblique |
| `phvron` | Helvetica-Narrow-Oblique |
| `phvrrn` | Helvetica-Narrow |
| `pncb` | NewCenturySchlbk-Bold |
| `pncbi` | NewCenturySchlbk-BoldItalic |
| `pncri` | NewCenturySchlbk-Italic |
| `pncr` | NewCenturySchlbk |
| `pncrc` | NewCenturySchlbk (Small Caps) |
| `pplb` | Palatino-Bold |
| `pplbi` | Palatino-BoldItalic |
| `pplbu` | Palatino-BoldUnslanted |
| `pplrrn` | Palatino-Narrow |
| `pplrre` | Palatino-Expanded |
| `pplri` | Palatino-Italic |
| `pplr` | Palatino |
| `pplro` | Palatino-Oblique |
| `pplru` | Palatino-Unslanted |
| `pplrc` | Palatino (Small Caps) |
| `psyr` | Symbol* |
| `psyro` | Symbol-Oblique* |
| `ptmb` | Times-Bold |
| `ptmbi` | Times-BoldItalic |
| `ptmrrn` | Times-Narrow |
| `ptmrre` | Times-Expanded |
| `ptmri` | Times-Italic |
| `ptmro` | Times-Oblique |
| `ptmr` | Times-Roman |
| `ptmrc` | Times-Roman (Small Caps) |
| `pzcmi` | ZapfChancery-MediumItalic |
| `pzdr` | ZapfDingbats* |

## 6.5 Expansion

This is a list of the possible expansions, in order from narrowest to widest.

| | | | |
|---|---|---|---|
| `o` | extra condensed | `x` | extended (by hand) |
| `c` | condensed (by hand) | `e` | expanded (automatic) |
| `n` | narrow (automatic) | `w` | wide |
| `r` | regular, normal, medium (usually omitted) | | |

Expansion of fonts is sometimes done automatically (as in PostScript **scale**), and sometimes done by humans. I chose 'narrow' and 'expanded' to imply the former, and 'condensed' and 'extended' to imply the latter, as I believe this reflects common usage.

## 6.6 Naming Virtual Fonts

In concert with releasing TEX 3.0 and METAFONT 2.7, Don Knuth wrote two new utility programs: `VFtoVP` and `VPtoVF`, which convert to and from "virtual" fonts. Virtual fonts provide a general interface between the writers of TEX macros and font suppliers. In general, therefore, it is impossible to come up with a general scheme for naming virtual fonts, since each virtual font is an individual creation, possibly bringing together many unrelated fonts.

Nevertheless, one common case is to use virtual fonts to map TEX's default accent and other character code conventions onto a vendor-supplied font. For example, `dvips` does this for fonts given in the PostScript "standard encoding". In this case, each font consists of a "virtual" `tfm` file, which is what TEX uses, a "raw" `tfm` file, which corresponds to the actual device font, and a `vf` file, which describes the relationship between the two.

This adds another dimension to the space of font names, namely, "virtualness" (or rather, "rawness", since it is the virtual `tfm` files that the users want to see). But we have already used up all eight characters in the font names.

The best solution I have been able to think of is this: prepend `r` to the raw `tfm` files; the virtual `tfm` files should be named with the usual foundry prefix. For example, the virtual Times-Roman tfm file is named `ptmr`, as usual; the raw Times-Roman `tfm` file is named `rptmr`. To prevent intolerable confusion, I promise never to give a foundry the letter `r`.

This scheme will work only as long as the virtualized fonts do not have design sizes; if they do, another foundry letter will have to be allocated, it seems to me.

A pox upon the houses of those who decided on fixed-length filenames!

## 6.3 Weight

This is a list of the possible weights, roughly in order from lightest to heaviest.

| | | | |
|---|---|---|---|
| `a` | hairline | `d` | demi |
| `t` | thin | `s` | semi |
| `i` | extra light | `b` | bold |
| `l` | light | `x` | extra bold |
| `k` | book | `h` | heavy |
| `r` | regular | `c` | black |
| `m` | medium | `u` | ultra |

## 6.4 Variants

The variants are:

| | | | |
|---|---|---|---|
| `a` | alternate | `n` | informal |
| `b` | bright | `o` | oblique (i.e., slanted) |
| `c` | small caps | `r` | normal (roman or sans) |
| `e` | engraved | `s` | sans serif |
| `g` | grooved (as in the IBM logo) | `t` | typewriter |
| `h` | shadow | `u` | unslanted italic |
| `i` | (text) italic | `x` | expert |
| `l` | outline | | |

If the variant is `r`, and the expansion is also normal, both the variant and the expansion are omitted. When the normal version of the typeface is sans serif (e.g., Helvetica), `r` should be used, not `s`. Use `s` only when the typeface family has both serif and sans serif variants. The "alternate" variant (`a`) is used by some Adobe fonts that have spiffy swashes and additional ligatures. The "expert" variant (`x`) is also used by some Adobe fonts with oldstyle figures and small caps.

Some fonts have multiple variants; Stone Informal Italic, for example. The only reasonable approach to these is to list all the letters for all the variants, choosing one to end with that is not also an expansion letter. Of course, it is possible that the name will become too long if you do this, but ... well, I'm open to suggestions. It's also possible the name will be ambiguous, if some new letter is used for expansions in the future. You can avoid this problem by adding the expansion `r` (if it doesn't make the name too long), and never using `r` for the last variant.

article for various addresses), so I can update the lists. You can get the most up-to-date version of these lists electronically, by anonymous ftp from the host `ftp.cs.umb.edu`. I will also send them to you by electronic mail, if necessary.

I give the letters in lowercase, which is preferred on systems where case is significant. Most lists are in alphabetical order by the abbreviations.

## 6.1 Foundry

This is the current list of foundries.

| | |
|---|---|
| `a` | Autologic |
| `b` | Bitstream |
| `c` | Agfa-Compugraphic |
| `g` | Free Software Foundation (`g` for GNU) |
| `h` | Bigelow & Holmes (with apologies to Chuck) |
| `i` | International Typeface Corporation |
| `p` | Adobe (`p` for PostScript) |
| `r` | reserved for use with virtual fonts; see below |
| `s` | Sun |

## 6.2 Typeface Families

The list of typefaces is:

| | |
|---|---|
| `ad` | Adobe Garamond |
| `ag` | Avant Garde |
| `ao` | Antique Olive |
| `at` | American Typewriter |
| `bb` | Bembo |
| `bd` | Bodoni |
| `bg` | Benguiat |
| `bk` | Bookman |
| `bl` | Balloon |
| `bv` | Baskerville |
| `bw` | Broadway |
| `cb` | Cooper Black |
| `cl` | Cloister |
| `cr` | Courier |
| `cn` | Century |
| `cs` | Century Schoolbook |
| `hv` | Helvetica |
| `gm` | Garamond |
| `go` | Goudy Oldstyle |
| `gs` | Gill Sans |
| `jo` | Joanna |
| `lc` | Lucida |
| `lt` | Lutetia |
| `nc` | New Century Schoolbook |
| `op` | Optima |
| `pl` | Palatino |
| `pp` | Perpetua |
| `rw` | Rockwell |
| `st` | Stone |
| `sy` | Symbol |
| `tm` | Times |
| `un` | Univers |
| `uy` | University |
| `zc` | Zapf Chancery |
| `zd` | Zapf Dingbats |

- Most typefaces are offered by several vendors. The version offered by vendor A is not compatible with that of vendor B.

- Typefaces typically come in different weights (hairline to extra heavy), different expansions (ultra condensed to wide), and an open-ended range of variants (italic, sans serif, typewriter, shadow, ...). No accepted standards exist for any of these qualities, nor are any standards ever likely to gain acceptance.

- The Computer Modern typeface family preserves traditional typesetting practice in at least one important respect: different sizes of the same font are not scaled linearly. This is in contrast to most commercial fonts available.

  Here is how I propose to divide up the eight characters:

  `FTTWVEDD`

where

- `F` represents the foundry that produced the font, and is omitted if there isn't one.

- `TT` represents the typeface name.

- `W` represents the weight.

- `V` represents the variant, and is omitted if both it and the expansion are "normal".

- `E` represents the expansion, and is omitted if it is "normal".

- `DD` represents the design size, and is omitted if the font is linearly scaled from a single `tfm` file.

  See the section on virtual fonts (towards the end) for an exception to the above.

The weight, variant, and expansion are probably all best taken from the original source of the typeface, instead of trying to relate them to some external standard.

Before giving the lists of abbreviations, let me point out two problems, to neither of which I have a good solution. 1) Assuming that only the English letters are used, two letters is enough for only 676 typeface families (even assuming we want to use all possible combinations, which is doubtful). There are many more than 676 typeface families in the world. 2) Fonts with design sizes over 100 pt are not common, but neither are they unheard of.

On to the specifics of the lists. If you adopt this proposal at your own installation, and find that you have fonts with some property I missed, please write to me (see the end of the

TeX uses for the font and is also the name of the `tfm` file. If there is another name word, that name is used as the PostScript name; if there is only one name word, it is used for both the TeX name and the PostScript name.

Note that `dvips` no longer registers the full PostScript name if an alias is given, so the single line

```
rptmr Times-Roman
```

would only allow `dvips` to find the `rptmr` font and not the `Times-Roman` font.

Incidentally, many font files (e.g., in the Adobe Type Library) include more characters internally than you can access directly without making a few changes. These characters typically have code $-1$ in the `afm` file and no corresponding 'constructed character' (`CC`) specification. For example, in Adobe Garamond there are Eth and eth, Thorn and thorn, plus things like copyright and onehalf, plus 58 accented letters Aacute thru zcaron (with accents over caps much better looking than you can get from constructed accents). To use these characters, give them character codes between 0 and 255 in the `afm` file, and build the corresponding EncodingVector (instead of StandardEncoding) in the `pfb` file. Then `afm2tfm` will generate a `vpl` file that includes accented characters as ligatures, as explained above; for example, 'Aacute' will be accessible as 'acute' followed by 'A' (as well as by its character code in the virtual font). A good sample encoding vector is provided by Donald E. Knuth in the file `AGaramond-Demo.pfb`.

# 6. Font Naming Conventions

This section of the manual has been written by Karl Berry and specifies a standard for naming of fonts for TeX. This standard has been adopted in `dvips`, and it is recommended that it be followed where possible.

As more typeface families become available for use with TeX, the need for a consistent, rational naming scheme for the font filenames concomitantly grows. Some (electronic) discussion has gone into the following proposal; I felt it was appropriate now to bring it before a wider community. In some respects, it follows and simplifies Mittelbach and Schöpf's article in *TUGboat*, volume 11, number 2 (June 1990).

Here are some facts about fonts that went into the hopper when creating this proposal:

- TeX runs on virtually all computers, under almost as many operating systems, all with their own idea of how files should be named. Any proposal regarding filenames, therefore, must cater to the lowest common denominator. That seems to be eight characters in length, not counting any extension, and with case being insignificant. Characters other than letters and numerals are probably unusable.

## 5.3 Non-Resident PostScript Fonts

If you want to use a non-printer-resident PostScript font for which you have a `pfb` file (an Adobe Type 1 font program), you can make it act like a resident font by putting a '`<`' sign and the name of the `pfb` file just after the font name in the `psfonts.map` file entry. For example,

```
StoneInformal <StoneInformal.pfb
```

will cause `dvips` to include `StoneInformal.pfb` in your document as if it were a header file, whenever the pseudo-resident font StoneInformal is used in a document. Similarly, you can generate transformed fonts and include lines like

```
StoneInformal-Condensed <StoneInformal.pfb "/StoneInformal .8 ExtendFont"
```

in `psfonts.map`, in which case `StoneInformal.pfb` will be loaded whenever StoneInformal-Condensed is used. (Each header file is loaded at most once per document. The `pfb` files should be installed in the `dvips` configuration directory [usually `/usr/lib/tex/ps`] with the other header files.)

When using such files, `dvips` is smart enough to unpack the standard binary `pfb` format into ASCII so there is no need to perform this conversion yourself.

## 5.4 Font Aliases

Some systems don't handle files with long names well—MS-DOS is a notable example. For this reason, `dvips` will accept an alias for such fonts. Such an alias should be the first word in the `psfonts.map` line. For instance, if we wanted the name `rptmr` to be used for the raw `Times-Roman` since our computer can't handle long names or, alternatively, we want to follow the standard naming conventions, we would use the following line in our `psfonts.map` file:

```
rptmr Times-Roman
```

The `tfm` file must have the name `rptmr.tfm`.

The distribution file `adobe` contains a list of the short names that should be used for most Adobe fonts currently available. Please reference this file when installing a new font and use the standard name.

The parsing of the `psfonts.map` file should be explained to eliminate all confusion. Each line is separated into words, where words are separated by spaces or tabs. If a word begins with a double quote, it extends until the next double quote or the end of the line. If a word starts with a less than character, it is treated as a font header file (or a downloaded PostScript font). If a word starts with a double quote, it is a special instruction for generating that font. Otherwise it is a name. The first such name is always the name

which create `ptmro.vpl` and `rptmro.tfm`. To use this, proceed as above but put the line

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

into `psfonts.map`. Then `rptmro` (our name for an obliqued Times) will act as if it were a resident font, although it is actually constructed from Times-Roman by PostScript hackery. (This oblique version of Times-Roman is obtained by slanting everything 1/6 to the right.) Similarly, you can get an expanded font by

```
localhost> afm2tfm Times-Extended -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

```
localhost> afm2tfm Times-Roman -V ptmrc ptmr
```

This is done strictly with a virtual font, however. In addition, the font on which the small caps font is based (in this case `ptmr` must already be created and installed. After running `afm2tfm` and `vptovf`, only `ptmrc.vf` and `ptmrc.tfm` should be installed; the created `ptmr.tfm` should not be used. If you create a slanted small caps font, the unslanted base font should already be created with the same parameters you give to create the slanted font, and you should give the name of the slanted font as the base name (last parameter) to `afm2tfm`.

The command line switches to `afm2tfm` are:

`-e` *ratio* All characters are stretched horizontally by the stated ratio; if it is less than 1.0, you get a condensed font.

`-s` *slant* All characters are slanted to the right by the stated slant; if it is negative, the letters slope to the left (or they might be upright if you start with an italic font).

`-v` *file* Generate a virtual property list `vpl` file as well as a `tfm` file.

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

guillemots). It also allows you to type 58 accented letters from 'Aacute' to 'zcaron' as if they were ligatures—namely as 'acute' followed by 'A', etc., where you can get 'acute' (`\char 19`) by defining a suitable control sequence. (The 58 constructed characters are found in the `afm` file, and constructed as specified by Adobe; you may want to adjust the spacing and/or add additional accent combinations by editing the `vpl` file.) 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).†

When `dvips` goes to use a font, it first checks to see if it is one of the fonts listed in a file called `psfonts.map`. If it is mentioned in that file, then it is assumed that the font is a resident PostScript font and can be found with the PostScript `findfont` operator. This file resides by default in `/usr/lib/tex/ps`, and consists of a single font per line. Note that only the raw PostScript font names should be listed in this file; the `vf` fonts should not be, since they are automatically mapped to the raw PostScript fonts by the virtual font machinery. The supplied `psfonts.map` file defines the most common PostScript fonts.

As much as possible, the PostScript fonts follow plain TEX conventions for accents. The two exceptions to this are the Hungarian umlaut (which is at position `0x7D` in `cmr10`, but position `0xCD` in `ptmr`) and the dot accent (at positions `0x5F` and `0xC7`, respectively). 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. Note that these definitions will not work with normal TEX fonts for the relevant accents; note also that these definitions are already part of the distributed `psfonts.sty`. In addition, the `\AA` that is used to typeset the Å character must be modified as shown.

```
\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}
```

These PostScript fonts can be scaled to any size. Go wild! Note, however, that using PostScript fonts does use up a great deal of the printer's virtual memory and it does take time. You may find downloading the Computer Modern fonts to be faster than using the built-in PostScript fonts.

## 5.2 Special Effects

Special effects are also obtainable, with commands such as

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

† See Donald E. Knuth, *TUGboat* v. 11, no. 1, Apr. 1990, pp. 13–23. "Virtual Fonts: More Fun for Grand Wizards."

This file is 'raw' because it does no character remapping; it simply converts the character information on a one-to-one basis to TEX characters *with the same code.*

In the following examples, we will use the font Times-Roman to illustrate the conversion process. For the standard 35 LaserWriter fonts, however, it is highly recommended that you use the supplied `tfm` and `vf` files that come with `dvips` (usually in a file called `dvipslib.tar.Z`, as these files contain some additional changes that make them work better with TEX than they otherwise would.

PostScript fonts have a different encoding scheme from that of plain TEX. Although both schemes are based on ASCII, special characters such as ligatures and accents are handled quite differently. Therefore we obtain best results by using a 'virtual font' interface, which makes TEX act as if the PostScript font had a standard TEX encoding. Such a virtual font can be obtained, for example, by the command

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

This produces two outputs, namely the 'virtual property list' file `ptmr.vpl`, and the TEX font metric file `rptmr.tfm`. The latter file describes an 'actual font' on which the virtual font is based.

To use the font in TEX, you should first run

```
localhost> vptovf ptmr.vpl ptmr.vf ptmr.tfm
```

and then install the virtual font file `ptmr.vf` in the virtual font directory (by default, `/usr/lib/tex/fonts/vf`) and install `ptmr.tfm` and `rptmr.tfm` in the directory for TEX font metrics (by default, `/usr/lib/tex/fonts/tfm`). (This probably has already been done for you for the most common PostScript fonts.) 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. Once this has been done, you're all set. You can use code like this in TEX henceforth:

```
\font\myfont=ptmr at 10pt
\myfont Hello, I am being typeset in Times-Roman.
```

Note that there are two fonts, one actual ('rptmr', which is analogous to a raw piece of hardware) and one virtual ('ptmr', which has typesetting know-how added). You could also say

```
\font\TR=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 called `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

## 4.7 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 often reduce them by more than a factor of five. For this reason, if the filename parameter to one of the graphics inclusion techniques starts with a backtick (`), the filename is instead interpreted as a command to execute that will send the actual file to standard output. Thus,

```
\special{psfile="`zcat foo.ps.Z"}
```

will include the uncompressed version of `foo.ps`. Since such a command is not accessible to TEX, if you use this facility with the `EPSF` macros, you need to supply the bounding box parameter yourself, as in

```
\epsffile[72 72 540 720]{"`zcat screendump.ps.Z"}
```

to include `screendump.ps`. Of course, the commands to be executed can be anything, including using a file conversion utility such as `tek2ps` or whatever is appropriate.

This extension is not portable to other `dvi2ps` programs. Yet.

# 5. Using PostScript Fonts

Thanks to Donald E. Knuth, the `dvips` driver now supports PostScript fonts through the virtual font capability. PostScript fonts are (or should be) accompanied by a font metric file such as `Times-Roman.afm`, which describes characteristics of a font called Times-Roman. To use such fonts with TEX, we need `tfm` files that contain similar information. These can be generated from `afm` files by running the program `afm2tfm`, supplied with `dvips`. This program also creates virtual fonts with which you can use normal plain TEX conventions.

Note that PostScript fonts tend to use a great deal of printer virtual memory, usually much more than the TEX bitmap fonts. This is because the information for the bitmapped fonts is stored in a very dense form and only downloads information for characters that are actually used, while modifying PostScript fonts for use by TEX (by creating a new width vector, among other things) takes a fair bit of PostScript memory.

## 5.1 The afm2tfm Program

The `afm2tfm` program can convert an Adobe `afm` file into a 'raw' TEX `tfm` file with the command

```
localhost> afm2tfm Times-Roman rptmr
```

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

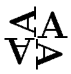
(with longer forms taking precedence over shorter forms, when they are used). An exception is the command

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

which will copy the commands from `filename` verbatim into the output (but 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 90 degrees.

To finish off this section, the following examples 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 when using the above forms. Make sure that each `gsave` on a page is matched with a `grestore` on the same page. Do not use `save` or `restore`. Use of these macros 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.

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

This will download the PostScript file called `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 keys 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

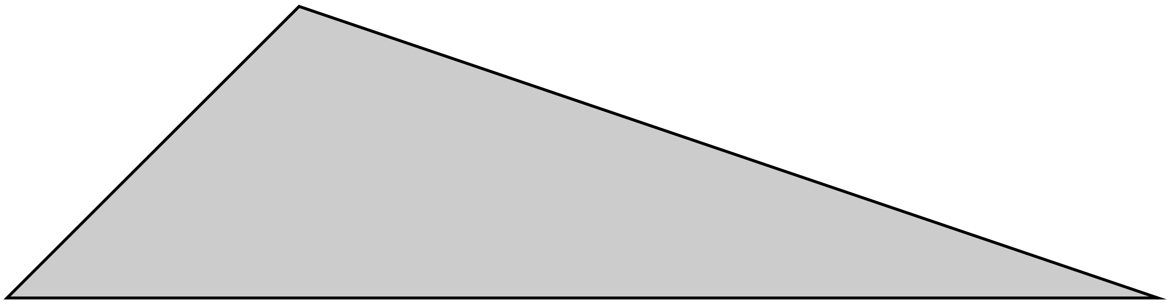
```
\special{/@scaleunit 1 def}
```

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

All of the methods for including graphics we have described so far enclose the graphic in a PostScript save/restore pair, guaranteeing that the figure will have no effect on the rest of the document. Another type of special command allows literal PostScript instructions to be inserted without enclosing them in this protective shield; users of this feature are supposed to understand what they are doing (and they shouldn't change the PostScript graphics state unless they 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:

## 4.4 Literal PostScript

For simple graphics, or just for experimentation, literal PostScript graphics can be included. Simply use a special command that starts with a double quote ("). 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}}
```

(Note that you are responsible for leaving space for such literal graphics.) Literal graphics are discouraged because of their nonportability.

## 4.5 Literal Headers

Similarly, you can define your own macros for use in such literal graphics through the use of literal macros. Literal macros are defined just like literal graphics, only you begin the special with an exclamation point instead of a double quote. These literal macros are included as part of the header material in a special dictionary called `SDict`. This dictionary is the first one on the PostScript dictionary stack when any PostScript graphic is included, whether by literal inclusion or through the `epsffile` macros.

## 4.6 Other Graphics Support

There are other ways to include graphics with `dvips`. One is to use an existing package, such as `emtex`, `psfig`, `tpic`, or `METAPOST`, all supported by `dvips`.

Other facilities are available for historical reasons, but their use is discouraged, in hope that some 'sane' form of PostScript inclusion shall become standard. Note that the main advantage of the `epsffile` macros is that they can be adapted for whatever form of special eventually becomes standard, and thus only minor modifications to that one file need to be made, rather than revising an entire library of TEX documents.

Most of these specials use a flexible key and value scheme:

Some common definitions are:

`\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.

`0pt` This definition forces natural sizes for all graphics by setting the width to zero, which turns on horizontal scaling.

`#1` This forces natural sizes too, by returning the first parameter only (the natural width) and setting the width to it.

`\hsize` This forces all graphics to be scaled so they are as wide as the current horizontal size. (In LaTEX, use `textwidth` instead of `hsize`.)

`0.5#1` This scales all figures to half of their natural size.

`\ifnum#1>\hsize\hsize\else#1\fi` This keeps graphics at their natural size, unless the width would be wider than the current `hsize`, in which case the graphic is scaled down to `hsize`.

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
```

## 4.3 Header Files

Often in order to get a particular graphic file to work, a certain header file might need to be sent first. Sometimes this is even desirable, since the size of the header macros can dominate the size of certain PostScript graphics files. The `dvips` program provides support for this with the `header=` special command. For instance, to ensure that `foo.ps` gets downloaded as part of the header material, the following command should be added to the TEX file:

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

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

`epsffile` where the size is given in TEX dimensions, this scaling will produce the correct, or expected, results. For compatibility with old PostScript drivers, it is possible to turn this scaling off 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 used in a `dvi` document, 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.

You can enlarge or reduce the figure by putting

```
\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. Alternatively you can force the vertical size to a particular size with

```
\epsfysize=<dimen>
```

in which case the height will be set and the width will be scaled proportionally. If you set both, the aspect ratio of the included graphic will be distorted but both size specifications will be honored.

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
```

A more general facility for sizing is available by defining the `epsfsize` macro. You can redefine this macro to do almost anything. This TEX macro is passed two parameters by `epsffile`. The first parameter is the natural horizontal size of the PostScript graphic, and the second parameter is the natural vertical size. This macro is responsible for returning the desired horizontal size of the graph (the same as assigning `epsfxsize` above).

In the definitions given below, only the body is given; it should be inserted in

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

`%%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 use them when you import the graphic.

If the document does not have such a bounding box, or if the bounding box is given at the end of the document, please complain to the authors of the software package that generated the file; without such a line, including PostScript graphics can be tedious.

## 4.2 Using the EPSF Macros

Now you are ready to include the graphic into a TEX file. Simply add to the top of your TEX file a line like

```
\input epsf
```

(or, if your document is in LaTEX or SliTEX, add the `epsf` style option, as was done to the following line).

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

This only needs to be done once, no matter how many figures you plan to include. Now, at the point you want to include the file, enter a line such as

```
\epsffile{foo.ps}
```

If you are using LaTEX or SliTEX, you may need to add a `leavevmode` command immediately before the `epsffile` command to get certain environments to work correctly. If your file did not (or does not currently) have a bounding box comment, you should supply those numbers you wrote down as in the following example:

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

(in the same order they would have been in a normal bounding box comment). Now, save your changes and run TEX and `dvips`; the output should have your graphic positioned at precisely the point you indicated, with the proper amount of space reserved.

The effect of the `epsffile` macro is to typeset 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 a 'natural' height. The graphic will be scaled by any `dvi` magnification in effect at the time.

Any PostScript graphics included by any method in this document (except `bop-hook` and its ilk) are scaled by the current `dvi` magnification. For graphics included with

# 4. Including PostScript Graphics

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. The most important feature of a good PostScript file—from the standpoint of including it in another document—is an accurate bounding box comment.

## 4.1 The 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 in terms of the lower left and upper right corners of a box just enclosing the graphic, and is thus referred to as a bounding box. These coordinates are given in PostScript units (there are precisely 72 PostScript units to the inch) 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 is standard ASCII, so you can use any text editor to do this.) If within the first few dozen lines there is a line of the form

```
%%BoundingBox: 0 1 2 3
```

(with any 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 (that `dvips` needs to position the graphic) is at the end of the file and should be moved to the position of the line above. This can be done with that same text editor, or with a simple Perl script.

If the document lacks a bounding box altogether, one can easily be added. Simply print the file. Now, take a ruler, and make the following measurements. All measurements should be in PostScript units, so measure it in inches and multiply by 72. Alternatively, the `bbfig` program distributed with `dvips` in the `contrib` directory can be used to automate this process.

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.

Now, 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.)

# 3. Paper Size and Landscape Mode

Most TeX documents at a particular site are designed to use the standard paper size (for example, letter size in the United States or A4 in Europe.) The `dvips` program defaults to these paper sizes and can be customized for the defaults at each site or on each 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 'normal' orientation is portrait). Alternatively, a document might be designed for ledger or A3 paper.

Since the intended paper size is a document design decision, and not a decision that is made at printing time, 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.

The format of the `papersize` special is

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

where the dimensions given above are for a standard letter sheet. The first dimension given is the horizontal size of the page, and the second 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 landscape document, the `papersize` comand would be given as

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

An alternate specification of `landscape` is to have a special of the form

```
\special{landscape}
```

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

Of course, using such a command only informs `dvips` of the desired paper size; you must still adjust the `hsize` and `vsize` in your TeX document to actually use the full page.

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

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 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. The `dvips` driver 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.

# 2. Using dvips

To use `dvips`, simply type

```
localhost> dvips foo
```

where `foo.dvi` is the output of TEX that you want to print. If `dvips` has been installed correctly, the document should come 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. The next time that document is printed, these fonts will already exist, so printing will go much faster.

Many options are available; they are described in a later section. For a brief summary of available options, just type

```
localhost> dvips
```

# DVIPS: A TEX Driver

Tomas Rokicki, rokicki@cs.stanford.edu

[This manual describes version 5.482.]

The dvips program converts a TEX dvi file into a PostScript file for printing or distribution. Seldom has such a seemingly easy programming task required so much effort.

## 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. Installation is covered in section 14 near the end of this document.

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 resolutions other than 300 dpi are also supported, even if they have a different resolution in the horizontal and vertical directions. High resolution output is supported for typesetters, including an option that compresses the bitmapped 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 æsthetics of the resulting output.