Table of Contents

Appendix B Program Index

Appendix A Concept Index

5 Ascii Translation

These sections describe various programs which can generate plain ascii from some file formats.

5.1 from TEX

Lametex will generate ascii from TEX source. It is available as:
`sunsite.unc.edu:/pub/Linux/apps/tex/lametex.tar.z`

LaTEX is used by the Linux Doc Projext to generate text versions of their manuals. I don't know where to find it.

5.2 from dvi

`dvi2tty` is a program which will process dvi into text. Aparently, it will also make an effort at reproducing graphics as well. I do not know where to find it.

5.3 from PostScript

Ghostscript 2.6.1 comes with a script file which will use `gs` to extract just the text from a ps file, called `ps2ascii`. (see Section 2.3 [PostScript], page 3, for information above for where it can be found). Further documentation is in the Ghostscript 2.6.1 distribution files `gs_2asc.ps` and `use.doc`

5.4 from troff

`groff -Tascii` or `-Tlatin1` ...

5.5 from ascii/latin1

The GNU program `recode` handles conversion between various forms of straight text encoding, ie from Latin-1 to ASCII. This is available on prep.ai.mit.edu.

`ws105.zfn.uni-bremen.de:/pub/gs261-svgalib.tar.gz`

Another possibly different svgalib patch is found in:
`ftp.cdrom.com:/pub/linux/misc`

The plain vgalib driver is available on Sunsite.

Texmgr is a program which will preview dvi under MGR. I don't beleive that it currently works under Linux MGR, but if it does, MGR uses sufficiently less memory and disk that this might be an attractive option for some.

dvgt is a program which will preview dvi with Linux svgalib, or on one of several types of graphics terminals including vt, tek, or a PC with MS-Kermit. It is available on sunsite.

4.3 xdvi

A beautifully legible previewing program for dvi with a handy zoom+pan feature. Will not interpret PostScript specials, which are understood only by `dvips` (back to the compiler, object file, and now linker analogy :-) To view a file, do '`xdvi file.dvi`'. This comes with either TEX or X in most distributions. Either way, you've probably got one. If not, look in `ftp.x.org:/contrib/`.

4.4 xtex

Xtex is similar in purpose to xdvi. I have tried to build it under Linux and failed. It is available as:
`ftp.x.org:/contrib/xtex-2.18.5.tar.z`

4.5 gxditview

Ditview produces a preview version of `troff` source using X fonts. '`groff -TX100 -mandoc` *man page*' will run gxditview to show you a typeset version of the man page. '`-TX75`' is the same thing, but tiny. Most distributions don't have a working one at all. A good one comes with the source to `groff`, which you might want to get anyway for the additional drivers (some distributions are missing some, including PostScript).
`prep.ai.mit.edu:/pub/gnu/groff-`*xxxx*`.tar.z`

4.6 non-X previewing

Ghostscript comes with pc video hardware drivers, but under un*x these are not a good thing. However, there are `gs` binaries around which will use the Linux VGA library (`svgalib`). The Ghostscript device for this is called linux, thus '`gs -sDEVICE=linux file.ps`' will show you an image of the PostScript. The environment variable `GSVGAMODE` is important for this. Set it to the nuber of the video mode you want, taken from the vga.h which comes with vgalib.

If you need this driver, a patch to put in Linux svgalib is available from the printing mail server or as:

```
ws105.zfn.uni-bremen.de:/pub/gs261-linuxdriver.sh
```

agw_____.pfb -Adobe-ITC Avant Garde Gothic-Book-r-normal–0-0-0-0-p-0-adobe-fontspecific
agwo____.pfb -Adobe-ITC Avant Garde Gothic-Book-o-normal–0-0-0-0-p-0-adobe-fontspecific
agd_____.pfb -Adobe-ITC Avant Garde Gothic-Demi-r-normal–0-0-0-0-p-0-adobe-fontspecific
agdo____.pfb -Adobe-ITC Avant Garde Gothic-Demi-o-normal–0-0-0-0-p-0-adobe-fontspecific
bkl_____.pfb -Adobe-ITC Bookman-Light-r-normal–0-0-0-0-p-0-adobe-fontspecific
bkli____.pfb -Adobe-ITC Bookman-Light-i-normal–0-0-0-0-p-0-adobe-fontspecific
bkd_____.pfb -Adobe-ITC Bookman-Demi-r-normal–0-0-0-0-p-0-adobe-fontspecific
bkdi____.pfb -Adobe-ITC Bookman-Demi-i-normal–0-0-0-0-p-0-adobe-fontspecific
com_____.pfb -Adobe-Courier-Medium-r-normal–0-0-0-0-m-0-adobe-fontspecific
coo_____.pfb -Adobe-Courier-Medium-o-normal–0-0-0-0-m-0-adobe-fontspecific
cob_____.pfb -Adobe-Courier-Bold-r-normal–0-0-0-0-m-0-adobe-fontspecific
cobo____.pfb -Adobe-Courier-Bold-o-normal–0-0-0-0-m-0-adobe-fontspecific
hv______.pfb -Adobe-Helvetica-Medium-r-normal–0-0-0-0-p-0-adobe-fontspecific
hvo_____.pfb -Adobe-Helvetica-Medium-o-normal–0-0-0-0-p-0-adobe-fontspecific
hvb_____.pfb -Adobe-Helvetica-Bold-r-normal–0-0-0-0-p-0-adobe-fontspecific
hvbo____.pfb -Adobe-Helvetica-Bold-o-normal–0-0-0-0-p-0-adobe-fontspecific
hvn_____.pfb -Adobe-Helvetica-Medium-r-Narrow–0-0-0-0-p-0-adobe-fontspecific
hvno____.pfb -Adobe-Helvetica-Medium-o-Narrow–0-0-0-0-p-0-adobe-fontspecific
hvnb____.pfb -Adobe-Helvetica-Bold-r-Narrow–0-0-0-0-p-0-adobe-fontspecific
hvnbo___.pfb -Adobe-Helvetica-Bold-o-Narrow–0-0-0-0-p-0-adobe-fontspecific
ncr_____.pfb -Adobe-New Century Schoolbook-Medium-r-normal–0-0-0-0-p-0-adobe-fontspecific
nci_____.pfb -Adobe-New Century Schoolbook-Medium-i-normal–0-0-0-0-p-0-adobe-fontspecific
ncb_____.pfb -Adobe-New Century Schoolbook-Bold-r-normal–0-0-0-0-p-0-adobe-fontspecific
ncbi____.pfb -Adobe-New Century Schoolbook-Bold-i-normal–0-0-0-0-p-0-adobe-fontspecific
por_____.pfb -Adobe-Palatino-Medium-r-normal–0-0-0-0-p-0-adobe-fontspecific
poi_____.pfb -Adobe-Palatino-Medium-i-normal–0-0-0-0-p-0-adobe-fontspecific
pob_____.pfb -Adobe-Palatino-Bold-r-normal–0-0-0-0-p-0-adobe-fontspecific
pobi____.pfb -Adobe-Palatino-Bold-i-normal–0-0-0-0-p-0-adobe-fontspecific
sy______.pfb -Adobe-Symbol-Medium-r-normal–0-0-0-0-p-0-adobe-fontspecific
tir_____.pfb -Adobe-Times-Medium-r-normal–0-0-0-0-p-0-adobe-fontspecific
tii_____.pfb -Adobe-Times-Medium-i-normal–0-0-0-0-p-0-adobe-fontspecific
tib_____.pfb -Adobe-Times-Bold-r-normal–0-0-0-0-p-0-adobe-fontspecific
tibi____.pfb -Adobe-Times-Bold-i-normal–0-0-0-0-p-0-adobe-fontspecific
zcmi____.pfb -Adobe-ITC Zapf Chancery-Medium-i-normal–0-0-0-0-p-0-adobe-fontspecific
zd______.pfb -Adobe-ITC Zapf Dingbats-Medium-r-normal–0-0-0-0-p-0-adobe-fontspecific

4.2 gspreview

This is another front-end for Ghostscript. I have gotten and built it, and actually preferred the user interface, but it had a few bugs. It didn't seem as full-featured as ghostview, though. (Not that there are all **that** many features in ghostview, but it does its job well). `ftp.x.org:/contrib/gspreview...`

2. `useScalableFonts(UseScalableFonts)`: Default value: true. This resource controls whether scalable fonts will be used. If you have an outline scaler in your X server, you should have this on. If you have an X terminal, you may get slightly better performance with this on. If you have to use the X11 bitmap scaler, turn this off. Fonts scaled by the bitmap scaler look worse than the default ghostscript fonts.
3. `logExternalFonts(LogExternalFonts)`: Default value: false. Controls whether to report when X11 fonts are being used.

The following fonts.scale makes all of the fonts of the Adobe Type Manager and Adobe Plus pack available in standard and ISO Latin 1 encoding. (We were able to purchase the above two packages at an educational discount price of $150.)

```
70
agw_____.pfb -Adobe-ITC Avant Garde Gothic-Book-r-normal–0-0-0-0-p-0-iso8859-1
agwo____.pfb -Adobe-ITC Avant Garde Gothic-Book-o-normal–0-0-0-0-p-0-iso8859-1
agd_____.pfb -Adobe-ITC Avant Garde Gothic-Demi-r-normal–0-0-0-0-p-0-iso8859-1
agdo____.pfb -Adobe-ITC Avant Garde Gothic-Demi-o-normal–0-0-0-0-p-0-iso8859-1
bkl_____.pfb -Adobe-ITC Bookman-Light-r-normal–0-0-0-0-p-0-iso8859-1
bkli____.pfb -Adobe-ITC Bookman-Light-i-normal–0-0-0-0-p-0-iso8859-1
bkd_____.pfb -Adobe-ITC Bookman-Demi-r-normal–0-0-0-0-p-0-iso8859-1
bkdi____.pfb -Adobe-ITC Bookman-Demi-i-normal–0-0-0-0-p-0-iso8859-1
com_____.pfb -Adobe-Courier-Medium-r-normal–0-0-0-0-m-0-iso8859-1
coo_____.pfb -Adobe-Courier-Medium-o-normal–0-0-0-0-m-0-iso8859-1
cob_____.pfb -Adobe-Courier-Bold-r-normal–0-0-0-0-m-0-iso8859-1
cobo____.pfb -Adobe-Courier-Bold-o-normal–0-0-0-0-m-0-iso8859-1
hv______.pfb -Adobe-Helvetica-Medium-r-normal–0-0-0-0-p-0-iso8859-1
hvo_____.pfb -Adobe-Helvetica-Medium-o-normal–0-0-0-0-p-0-iso8859-1
hvb_____.pfb -Adobe-Helvetica-Bold-r-normal–0-0-0-0-p-0-iso8859-1
hvbo____.pfb -Adobe-Helvetica-Bold-o-normal–0-0-0-0-p-0-iso8859-1
hvn_____.pfb -Adobe-Helvetica-Medium-r-Narrow–0-0-0-0-p-0-iso8859-1
hvno____.pfb -Adobe-Helvetica-Medium-o-Narrow–0-0-0-0-p-0-iso8859-1
hvnb____.pfb -Adobe-Helvetica-Bold-r-Narrow–0-0-0-0-p-0-iso8859-1
hvnbo___.pfb -Adobe-Helvetica-Bold-o-Narrow–0-0-0-0-p-0-iso8859-1
ncr_____.pfb -Adobe-New Century Schoolbook-Medium-r-normal–0-0-0-0-p-0-iso8859-1
nci_____.pfb -Adobe-New Century Schoolbook-Medium-i-normal–0-0-0-0-p-0-iso8859-1
ncb_____.pfb -Adobe-New Century Schoolbook-Bold-r-normal–0-0-0-0-p-0-iso8859-1
ncbi____.pfb -Adobe-New Century Schoolbook-Bold-i-normal–0-0-0-0-p-0-iso8859-1
por_____.pfb -Adobe-Palatino-Medium-r-normal–0-0-0-0-p-0-iso8859-1
poi_____.pfb -Adobe-Palatino-Medium-i-normal–0-0-0-0-p-0-iso8859-1
pob_____.pfb -Adobe-Palatino-Bold-r-normal–0-0-0-0-p-0-iso8859-1
pobi____.pfb -Adobe-Palatino-Bold-i-normal–0-0-0-0-p-0-iso8859-1
sy______.pfb -Adobe-Symbol-Medium-r-normal–0-0-0-0-p-0-iso8859-1
tir_____.pfb -Adobe-Times-Medium-r-normal–0-0-0-0-p-0-iso8859-1
tii_____.pfb -Adobe-Times-Medium-i-normal–0-0-0-0-p-0-iso8859-1
tib_____.pfb -Adobe-Times-Bold-r-normal–0-0-0-0-p-0-iso8859-1
tibi____.pfb -Adobe-Times-Bold-i-normal–0-0-0-0-p-0-iso8859-1
zcmi____.pfb -Adobe-ITC Zapf Chancery-Medium-i-normal–0-0-0-0-p-0-iso8859-1
zd______.pfb -Adobe-ITC Zapf Dingbats-Medium-r-normal–0-0-0-0-p-0-iso8859-1
```

I believe these values to be a good compromise between dynamic allocation and fall back onto a fairly good color cube for dithering.

You can use the foreground and background colors to accomplish "reverse video". However, if you have a grayscale device, it may be better to reverse the gray ramp using the following PostScript code fragment:

```
[{1 exch sub} /exec load currenttransfer /exec load] cvx settransfer
```

The X11 driver now supports native X11 fonts. If you have installed the HP XLFD font extensions into your font or X server. Ghostscript will also be able to use platform fonts at rotations of 90 degrees, with mirroring, and anamorphic scaling.

The X11 driver does most if its work silently. You can get it to report when it is using an X11 font by setting the logExternalFonts boolean in your X resources.

The X11 driver is setup to use the standard fonts distributed with X11R5. We purchased the Adobe Type Manager and the Adobe Plus Pack. These font packages give all the fonts normally found in the Apple LaserWriter Plus. The X11 driver is setup to handle these fonts as well. (They are a superset of the bitmap fonts distributed with X11.)

You may set the regularFonts, symbolFonts, or dinbatFonts resources if you have different fonts available. Each font name must have 7 dashes or it will be ignored. Minimize the use of wildcards to promote faster matching. (I once encountered an X server that took many seconds to do a font lookup when wildcards were carelessly used.)

There is a different list of fonts for each common encoding. Regular fonts may be accessed in standard or ISO Latin 1 encoding. The bdf files that are distributed with X11 are in the ISO Latin 1 encoding. This leaves out the ligatures. Luckily, the ligatures are present in the bdf files, but are not given an encoding, essentially commenting them out. You can use the `fixfont` program from the xproof distribution (`Ftp.Cs.Wisc.Edu:/Pub/X/Xproof.Tar.Z`, or `Ftp.X.Org:/Contrib/Xproof.Tar.Z`) to reencode the bdf files and build X11 fonts that contain the ligatures (i.e standard encoding).

If you have the Type1 fonts mentioned above, and you installed the Type1 rasterizer into you font or X server, you can use the appended fonts.scale to name your fonts so that ghostscript can find them.

Font resources:

1. `useExternalFonts(UseExternalFonts)`: Default value: true. This resource controls whether X11 fonts will be used.

The customization choices mentioned in make.doc have been moved into X resources and are now configured at run time rather than compile time. Sorry, this section of make.doc did not get revised for the 2.6.1 release.

If `useBackingPixmap` is set, ghostscript will attempt to allocate a backing pixmap. If one cannot be allocated, ghostscript will issue a warning and ask for backing store instead. (Since there were insufficient resources for a backing pixmap, the X server may not provide backing store either.)

Color Handling was totally revamped for gs 2.6.

Ghostscript first checks for a suitable standard colormap. If you have static colormap in your X server, it would be best to store a standard colormap property on the root window describing the color layout. Ghostscript will then be able to take full advantage of the device. If you have a standard colormap installed, ghostscript will start slightly faster since it does not have to allocate colors for a cube or ramp.

If no standard colormap is available, ghostscript will allocate an RGB cube or gray ramp. Ghostscript tries for a 5x5x5 cube on a color device, and a 128 gray ramp on grayscale devices. It will never ask for more than 1/2 of the colors for a RGB cube or gray ramp. It also takes into account the number of significant bits per pixel. (i.e. It won't ask for 128 gray levels if you only have 16 available.)

Ghostscript will attempt to allocate colors that are off the color cube/ramp as the picture is being rendered. Ghostscript will keep track of 256 dynamic colors. After all these are allocated, ghostscript asks the X server directly.

The foreground and background color can be set explicitly. This is important for the visually impaired and when using the ghostview widget.

Color Resources:

1. `palette(Palette)`: Default value: '`Color`'. Other allowable settings: '`Grayscale`', '`Monochrome`'. The palette resource is used to restrict the palette used for display. One can set palette to '`Grayscale`' or '`Monochrome`' to see how a file would be rendered in grayscale or monochrome on a color display. I use it to avoid dithering of gray- scale figures on a color display with 4-bit DACs.
2. `maxGrayRamp(MaxGrayRamp)`: Default value: 128. Maximum number of gray levels that ghostscript will attempt to allocate. (It won't try for more than this on an 8-bit pseudo color display even if you set it higher.) Set this lower if you want a smaller ramp and would prefer ghostscript to use dynamic colors.
3. `maxRGBCube(MaxRGBCube)`: Default value: 5. Maximum number of colors levels that ghostscript will attempt to allocate. (It won't try for more than this on an 8-bit pseudo color display even if you set it higher.) Set this lower if you want a smaller ramp and would prefer ghostscript to use dynamic colors.

4 Previewing

These sections describe various ways to preview things under Linux - that is, how to view them in a way approximating their final form without printing them out.

4.1 ghostview

Ghostview, a companion program for `gs`, previews PostScript on an X display. It also lets you select individual or ranges of pages from a PostScript document to print using `lpr`. The new version, 1.5, has fixed a few glitches which never bothered me but may make a difference to you. It also calls `gs` with the '`-dSAFER`' option and has a few more resource and command-line options relative to 1.4.1. The real installation is from:
`prep.ai.mit.edu:/pub/gnu/ghostview-`*xxx*`.tar.gz`

It builds out of the box. Ghostview requires `gs` to work. The new version of `gs`, 2.6.x, will use X display fonts in an effort to improve legibility at the low resolutions of a video monitor (a previous failing of this pair relative to commercial display-PostScript based systems). This works very well for me at least, at the expense of exact character positioning (X fonts have different widths). In fact, I thought that Ghostview looks better than Sun's pageview the other day when I looked at the same page in oth programs side-by-side. Ghostview/Ghostscript also has much more intelligent color handling than pageview. You might wish to let `gs` render some Type 1 fonts you install instead of using platform fonts (or the awful fonts `gs` comes with. To do this while in Ghostview (or in any situation involving the X11 driver), place '`ghostscript.useExternalFonts: false`' in your `.Xdefaults` file, and the platform fonts will not be used.

This is part of a message posted to gnu.ghostscript.bug by Tim Theisen <ghostview@cs.wisc.edu>:

> (note that the usual Linux X-server, XFree, is simply an enhanced version of MIT's effort at an i386 X-server (X386), and does contain the X11R5 Type 1 rasterizer which I beleive was contributed by IBM.)
>
> Ghostscript now uses the X Toolkit to pick up X Resources. Now ghostscript uses the standard X rules that allow more specific resources to override less specific ones giving users the full power of X resources to control the X11 driver. It also allows system administrators to establish an application defaults file with resources specific to their ghostscript installation.

```
# to print text out well at any size.  This one does not.  These options
# are also handy if you want to do your own snazzy header page, much
# like NeWSPrint from Sun does (although running PostScript through
# the display server to get it interpreted is a bit much :)
#
#
# gs will reset the printer anyway, so the this text setup doesn't matter.
# setup should include the escape code for \n conversion, if applicable.
#
printf "<printer setup for text printing (escape codes, etc)>"

read first_line
first_two_chars=`expr $first_line : '\(..\)'`

if [ "$first_two_chars" = "%!" ]; then # it's PostScript

        /usr/bin/gs -dSAFER -dNOPAUSE -q -sDEVICE=?????? -sOutputFile=- -

else # it's plain text

        echo -n $first_line
        cat
        printf "\014"

fi
```

Note that for the paranoid, shell scripts run as someone other than the user are sometimes a security hole, but this is not the case with lpd filters as the script's environment is not under the control of the potential cracker.

3.30 Magic Filters

Magic filters deduce their input files' types from 'magic numbers' (distictive byte patterns at particular offsets). Magic filters are usually perl scripts, shell scripts or C programs that simply identify the file type then call the appropriate non-magic filter. Blatent plug :-) Brian has a generic magic filter bash script that selects the right filter to use based on the output of the `file` command. With a suitable magic filter (and 3 associated non-magic filters) you can do things like:

```
lpr -d file1.dvi file2.div.Z file3.ps file4.texinfo.gz
```

(BTW confguring `lpr` to handle texinfo files is getting a bit silly).

This is now on the mailserver or at: `tsx-11.mit.edu:pub/linux/sources/usr.bin/magic-filter-0.4.tar.gz` (Although the release number will possibly change in future).

'`apsfilter`' is a rather easier to use shell script that requires no additional filters and which is pre-configured for HP compatible laser printers. This is also available on the mailserver.

An example written in C, which may be easily adapted to most installations is available from the printing mail server as '`lpr_if.c`'.

Magic filters should never specified as '`of`' as the output filter only gets called once if a number of files are printed without a gap. There are other more subtle problems too using '`of`'.

IMHO (Brian) magic filters as '`if`' are inelegant as they may prevent you, say, listing a PostScript or nroff file. (Most people disagree with me on this point.)

3.31 Magic Filter Examples

The following is an example of a magic shell script which should take either PostScript or text and deal with it:

```
#!/bin/sh
# This is based on a script I received from Scott Doty and which was
# written by Keith Walker.  Keith's script made use of the fact that
# lpd passes options to if:
#
#  <if> -w<width> -l<length> -i<indent> -n <user> -h <host> <accountingfile>
#
```

3.28 Filters from programs that won't read STDIN

Some of the programs that are used in writing `lpd` filters are not capable of taking their input from their standard input. For example `dvilj2p` insists on a named file as its input (and what's more expects one with a `.dvi` suffix) so do this:

```
#!/bin/sh
ln -s /proc/self/fd/0 /tmp/$$.dvi
dvilj2p /tmp/$$
rm /tmp/$$.dvi
```

Note: If it wasn't for the fact that `dvilj2p` adds a `.dvi` suffix you wouldn't need the temporary symlink and could just specify `/proc/self/fd/0` directly. People who use this trick often usually permanently '`ln -s /proc/self/fd/0 /dev/stdin`'. If you're highly security concious and don't allow access to `/proc` you'll need to create a temporary file.

3.29 Having many filters

Historically the `lpr` command was created to support a finite set of possible file types. You can, in fact, use any of the filters for any reason. If you're never going to use Benson Varian raster files you could use the '`-v`' switch for GIF files. You could even use '`-d`' for low res and `-v` for high res. Remember that if you create a filter for a file format that takes a long time to process then your printer may sit idle between print jobs even when there are things in the queue.

If you are on a network remember that the filter setups go on the print server. One way to avoid running out of filter options is to define several logical printers in `/etc/printcap` that all point to the same physical one and put each filter in the '`if`' field of a different printcap entry. This has the advantage that you can set the `PRINTER` enviroment variable to choose your filter rather than having to specify it on the command line each time. One small problem with this is that you have no control over the order in which files from separate queues are printed.

Another (and these days more common) way to avoid running out of possible types is to use magic filters.

A trick most people find useful when testing filters that make use of their command line arguments is to include '`echo $* >>/tmp/filter-log`' near the top of the script.

If the filter works when you test it but still doesn't work when called by `lpd` then you may have forgotten the '`#!/bin/sh`' header. You may also need to set PATH within the script since the daemon's PATH may not have everything you need. Note also that the filter is run with uid=daemon so any programs it calls sould be world executable.

3.26 Output ('`of`') filters

Never use these. (Well strictly speaking there are circumstances but you're unlikey to meet them). Recently (early '94) there has been a spate of preople on c.o.l.help advokating the use of output filters. Using '`of`' filters means that if a printout is queued while another is already printing the 2 will be run together with a form-feed between. Any printer initialisation or file type detection will therfore not be performed for the second file and it will probably be printed incorrectly. There are other more subtle ways in which output filters can do unexpected things. IMHO: If using an output filter is the answer, it was probably a silly question.

3.27 Getting filters for given printers

From: B.A.McCauley@bham.ac.uk (Brian McCauley)

Because writing a filter usually takes about 10 minutes once you've found the right program (`gs`, `dvilj` etc.) there's little call for ftp archives of printer filters but we are thinking of creating an extensive example file to go with this document.

If you already have a program to print, say, DVI on your printer by some mechanism then making it into a filter is usually a matter of writting trivial shell script See Section 3.24 [Writing lpd filters], page 43. If the program you are using insists on reading a names file as input see the next question. Text mode filters are trivial too (see this HOWTO) unless you want lpr to have a choice of fonts in which case they are slightly harder than trivial. You will probably want to insert and 'echo -ne' command at the beginning and end of your filter to set up the font etc to your liking.

3.24 Writing lpd filters

In normal Un*x terminology, filters are just programs (so they must have execute permission) that read a stream from their standard input and write to their standard output.

lpd filters are filters in the sense that thay read STDIN and write to STDOUT, but are unusual in that they may assume that their standard input is a random acess file file and may perform lseek() operations on it.

All lpd filters have a common command line syntax (or more often simply ignore command line parameters). For details of the command line parameters RTFM(printcap).

If you want to write a shell script filter it must have a #!/bin/sh (or perl or csh) header. Here is the generic form of a filter to accept PostScript.

```
#!/bin/sh
/path.../gs -q -dSAFER -dNOPAUSE -r??? -sDevice=?????? -sOutputFile=- -
```

Place the full pathname of the script as one of the filters (but not '`of`'!) parameter in the printcap for your printer. I suggest putting such scripts in `/usr/lib/lpd/`. It is also usual to keep filters in the spool directories but this goes against normal practice of keeping programs and data neatly apart. ('`-dSAFER`' attempts to protect against PostScript interpreter security holes, '`-q`' and '`-dNOPAUSE`' make it run nonstop, and Device is the appropriate special file for your printer).

Here is an Epson FX-80 dvi filter using ghostscript:

```
#!/bin/sh
/usr/TeX/bin/dvips -f | \
              /usr/bin/gs -q -dSAFER -sDEVICE=eps9high -r120x216 \
              -dNOPAUSE -sOutputFile=- -
```

More tools useful for making filters are described elsewhere in this document.

3.25 Debuging lpd filters

It's easier to debug filters if you test them in an immediate shell before you install them. (If your filter makes use of its command line arguments you'll have to specify them too). '`my-new-filter` <*file* `>/dev/lp1`'

3.23 Using `lpr` over a network

To print on the printer listed as `foo` in the printcap on machine `bar.baz.net` from the machine `mine.baz.net` you put an entry like this in your `/etc/printcap` (on `mine.baz.net`):

```
foo:lp=:rm=bar.baz.net:rp=foo:sd=/usr/lpd/spool/foo:
```

and, of course, create the spool directory `/usr/lpd/spool/foo`.

There's no point specifying filters and the like in `mine.baz.net:/etc/printcap` as it's the ones in `bar.baz.net:/etc/printcap` that will get used.

On the machine `bar.baz.foo`, you need to put `mine.baz.net` on a line by itself in either `/etc/hosts.equiv` or `/etc/hosts.lpd`; note that putting it in `/etc/hosts.equiv` will allow for unauthenticated logins as well as printing. `/etc/hosts.lpd` is for printing only.

The machines listed in `/etc/hosts.*` should be described canonical names or numbers as lpd starts with the IP address and performs a revervse DNS lookup to get the name. If you are not sure of cannonical name you can just list all the names you know for a machine. (If you have `dig` then the command '`dig -x` *a.b.c.d*' can be used to get the canonical name of IP address *a.b.c.d*.)

If the printer server is not nunning a BSD style spooler then it should still be possible to get it to work but the authority files may have a different names or formats. For example Chris Nystrom <chrisn@medianet.com> found that he had to create a file on the remote machine called `/usr/spool/lp/admins/lp/Systems` that listed his Linux box's name. We do not know if this is a SYSV thing or something exclusive to dynix/ptx 2.0.3 that he is using on his Sequent.

If you can't get remote printing to work thrugh lpd you may be able to simply use remote command exectution like this:

```
rsh bar.baz.net "lp -dlp" < file
```

This example would be for a remote system using a SYSV type printing system on host `bar.baz.net`.

```
for ($i=0; !($_ = $ARGV[$i]) || !/^-i([0-9])+/; $i++) {}
print pack("cAc",27,"l",$1);
while (<STDIN>) { print; }
```

3.21 Why 'lpr -p' doesn't work?

Because it's broken. `lpd` always thinks that the printer is 0 characters wide regardless of what `/etc/printcap` or the `lpr` arguemnts say. The lpd-FAQ contained a patch but it has now been applied to lpd-590p1 and later. (Appologies to anyone who wanted this patch after the lpd-FAQ merged with printing-how.to. It was dropped in the mistaken belief that a new release of lpd-590 was iminent).

One other thing: `lpd` calls `pr` by full pathname so if you keep `pr` somewhere different from `/usr/bin/pr` you will need a symlink. (Where `lpd` expects to find `pr` may vary from version to version).

3.22 lpc and lpq warning of missing daemons

One `lpd` process runs all the time and it spawns children to handle each printer as needed. The health of the master daemon is not explicity reported by `lpc` but the absence of errors indicates that it is healthy. See Section 3.5 [lpd not working], page 31. The '`lpc stat`' command will display the message "no daemon present" for each queue that is not actually printing at the time - this is completely normal. If printing has been disabled or the queue is empty then this is not an error condition. `lpq` is even more alarmist and will say "Warning: no daemon present". If the daemon is absent when the queue has entries and has not been explicitly stopped then this warning probably indicates an error in a filter. Fix the filter then use '`lpd up` *queue-name*' to restart it.

Sometimes when shutting down a printer `lpc` will get confused and try to kill a non existant daemon. This leads to irritating but harmless error messages. In lpd-590p2 these are much rarer.

font was last used on the printer then define the '`tr`' "capability" to be your printer's font reset command.

If you want a fancy customised banner (or have a PostScript printer) leave '`:sh:`' in the printcap and make each of your filters print the banner. All the information to put on the banner is included in the filter's positional parameters. RTFM(printcap) for details. [If you're using <B.A.McCauley@bham.ac.uk>'s magic-filter package then call the code to print the banners from the config script.]

3.18 Spooling text to a PostScript printer

You need a filter based on a program that converts ascii to PostScript. The most well known of these is `enscript` but it's also the hardest to find (being non-free). Others include `a2ps`, `nenscript`, and `mpage`. See Section 2.7 [Printing text via PostScript], page 9.

3.19 Why graphics files are sometines truncated

This is usually because you've got a limit set on the maximum size file that can sit in the spool queue. Put '`mx#0`' in your printcap.

3.20 Why '`lpr -i`' doesn't work

To get '`lpr -i`' to work you need a filter installed as '`if`' that implements it. The '`-i`' switch is simply passed on by `lpd` to the filter. The filter called `lpf` that comes with `lpd` supports this feature but can only be used to print text. If you whant to use this program but still want your filter to do some printer specific initialisation then write script thus:

```
#!/bin/sh
# My initialisation stuff goes here
exec /usr/lib/lpf $*
```

More reasonably you could have your filter script send the printer left margin sequence.

```
#!/usr/bin/perl
# This example is in perl for a change because converting numbers
# to characters is tricky in shell script
```

```
*** 1271,1277 ****
        }
  #ifdef LINUX
        ttybuf.c_cflag &= ~FC;          /* not quite right! */
!       ttybuf.c_cflag |= FS;           /* not quite right! */
  #else
        ttybuf.sg_flags &= ~FC;
        ttybuf.sg_flags |= FS;
--- 1271,1277 ----
        }
  #ifdef LINUX
        ttybuf.c_cflag &= ~FC;          /* not quite right! */
!       ttybuf.c_cflag |= ~FS;          /* not quite right! */
  #else
        ttybuf.sg_flags &= ~FC;
        ttybuf.sg_flags |= FS;
-------------------------------Cut Here------------------------------------
```

3.16 Printers that are not simple devices

[Firstly I'll explain the subject.] The most common example is a printer that is connected via a network in some strange way. For example consider a printer connected to a host with which you can only communicate via E-mail.

To use such a printer through `lpr` the '`lp`' capability of the print queue should be directed to a `/dev/null` type device (e.g. '`mknod /dev/null1 c 1 3`') but not `/dev/null` itself as `lpd` opens the device exclusively. Each filter must must explicitly uuencode and mail its output.

In more complex cases if you already have an '`if`' or '`of`' filter for a strangely connected printer then other filters can pass their output to/through this filter to avoid duplication of effort. In this case the '`if`' filter should usually be called with the '`-c`' switch to minimise the further manipulations if performs.

I've heard someone has had some success trying something like this with Novell NetWare and the free mail transfer agent "Charon".

3.17 Generating burst or banner pages

For a simple text printer (in particular not PostScript) and a simple text banner simply take '`:sh:`' out of the printcap record. If you want to prevent the banner comming out in whatever

...or if you didn't have an old '`if`' installed...

```
#!/bin/sh
echo if: $* >> /var/spool/lpd/results
cat > /dev/lp
# the ''echo -ne'' assumes that /bin/sh is realy bash
echo -en \\f > /dev/lp
```

Make sure it's world-executable and world-readable. Try out your script ('`/usr/lib/lpd/if < `*somefile*') and see if it prints.

3. Set the '`if`' capability in your printcap to call this script, e.g. '`if=/usr/lib/lpd/if`'
4. Use stty to correctly set your port settings. Try to print now. You should be able to tell if things are being spooled, and things **should** be printed, if your manual testing of the '`if`' script works. But this is a kludge, so the idea is not to use the '`if`' script.

Assuming the above method using the '`if`' filter works and that you believe that you have specified what you think are the correct flags and baud rate in printcap; check '`stty -a < /dev/ttyS2`' (or whatever your printer port is). If the settings are not correct, check your flags against your printout from termios.h. If the settings are **way** not correct, you may need a fixed lpd. The patch follows, and you can probably see why it's needed :-) Note: this patch is reversed and has already been applied (uh... unapplied :-)) to lpd-590p2 so don't apply it if you already have that version or later.

(the patch is coming in just a sec)

When I was setting mine up, I followed a sequence like this:

```
lprm whatever # (make sure queue is empty and lpd is running)
stty correct settings < /dev/ttyS2
lpr something small
stty -a < /dev/ttyS2  # (often had to ctrl-c out of this one)
```

twiddle with flags

```
lprm whatever # make sure queue is empty again...
```

Here's the patch (I it's reversed so apply it with '`-R`' - or, in practice, by hand!):

```
-------------------------------Cut Here-------------------------------------
*** lpd-590/lpd/printjob.c  Thu Jul  8 20:56:59 1993
--- lpd-590/lpd/printjob.c~ Sat Feb 27 09:07:01 1993
***************
```

```
/* c_cflag bit meaning */
#define CBAUD   0000017
```

This section lists the meaning of the 'fc#' and 'fs#' bits. You will notice that the names there (after the baud rates) match up with one of the lines of stty output. Didn't I say this was going to be easy?

Note which of those settings are preceded with a - in your stty output. Sum up all those numbers (they are octal). This represents the bits you want to **clear**, so the result is your 'fc#' capability. Of course, remember that you will be setting bits directly after you clear, so you can just use 'fc#0177777' (I do).

Now do the same for those settings (listed in this section) which do not have a - before them in your stty output. In my example the important ones are CS8 (0000060), HUPCL (0002000), and CREAD (0000200). Also note the flags for your baud rate – mine is 0000015. Add those all up, and in my example you get 0002275. This goes in your 'fs#' capability ('fs#02275' works fine in my example).

Do the same with set and clear for the next section of the include file, "c_lflag bits". In my case I didn't have to set anything, so I just use 'xc#0157777' and 'xs#0'.

Once your printcap is set up, try it out. If things don't work, see the next section.

3.15 cat works to the serial port, but not lpd (1)

Generally getting lpd up and running is explained elsewhere, but if you are having trouble with serial port settings you can prevent `lpd` from trying to configure your port by treating you printer as one that does not present a normal device interface. See Section 3.16 [Printers not in /dev], page 39.

1. Set your printer (in your printcap) to `/dev/null1`. ('`mknod /dev/null1 c 1 3`' because you don't want `/dev/null` to be opened exclusively).

   remove the baud rate and flags settings from your printcap.
2. Create a script such as this:

   ```
   #!/bin/sh
   echo if: $* >> /var/spool/lpd/results
   # /dev/lp is linked to /dev/ttyS2 which has the printer
   exec your-old-input-filter $* > /dev/lp
   ```

3.14 Printing with lpd to a serial port

The first if lpd complains about "ioctl(TIOCEXCL)" being unimplemented you need a version of lpd that doesn't care (eg. lpd-590p2)

There are two sets of flags which you will need to set, plus the baud rate (Note: the 'fc' flag setting seems to override the 'br#' capability, so be sure to set that correctly as well as the 'br#'!).

Each of the flags can have bits set and cleared. Clearing is done first, so specify the clear flags ('fc#' and 'xc#') before the set flags ('fs' and 'xs').

Setting the 'br#' capability is self-explanatory. Example: 'br#9600'

It is very easy to translate from `stty` settings to printcap flag settings. If you need to, see the man page for stty now.

Use stty to set up the printer port so that you can cat a file to it and have it print correctly. Here's what 'stty -a' looks like for my printer port:

```
dina:/usr/users/andy/work/lpd/lpd# stty -a < /dev/ttyS2
speed 9600 baud; rows 0; columns 0; line = 0;
intr = ^C; quit = ^\; erase = ^?; kill = ^U; eof = ^D; eol = <undef>;
eol2 = <undef>; start = ^Q; stop = ^S; susp = ^Z; rprnt = ^R; werase = ^W;
lnext = ^V; min = 1; time = 0;
-parenb -parodd cs8 hupcl -cstopb cread -clocal -crtscts
-ignbrk -brkint -ignpar -parmrk -inpck -istrip -inlcr
-igncr -icrnl ixon -ixoff -iuclc -ixany -imaxbel
-opost -olcuc -ocrnl -onlcr -onocr -onlret -ofill -ofdel nl0 cr0 tab0
bs0 vt0 ff0
-isig -icanon -iexten -echo -echoe -echok -echonl -noflsh -xcase -tostop
-echoprt -echoctl -echoke
```

The only changes between this and the way the port is initialized at bootup are `-clocal`, `-crtscts`, and `ixon`. Your port may well be different depending on how your printer does flow control.

Once you have your stty settings right, so that 'cat file > /dev/ttyS2' (in my case) sends the file to the printer, look at the file `/usr/src/linux/include/linux/termios.h`. This contains a lot of `#define`s and a few structs (You may wish to cat this file to the printer (you do have that working, right?) and use it as scratch paper – I (Andrew Tefft <teffta@engr.dnet.ge.com> did!). Go to the section that starts out

```
cat
echo -ne \\f
```

3.12 Resetting the printer between each file printed

Either make your filters do it or define the 'tr' "capability" in /etc/printcap to be your printer's font reset command. For details of the format of this string see the question on the format of printcap. This may not work if a printout crashes in the middle of an escape sequence - putting a lot of ^@ on the front may help but this probably won't be enough it you were printing raster graphics when the filter died.

3.13 Preventing formfeed after each file printed

If you don't have an 'if' specified in /etc/printcap then lpd will automatically put a formfeed at the end of each file. If you're using a filter then it's up to the filter to decide if it wants to put a formfeed. To disable formfeed completely if you don't have an 'if' put ':ff=:' in your /etc/printcap. But please note this suppresses the formfeed that would usually be printed if a filter dies. If you want formfeeds after text printouts but not on printouts printed with 'lpr -l' then create the following 'if' filter:

```
#!/bin/sh
# the ``echo -ne'' assumes that /bin/sh is really bash
cat
if [ "$1" != -c ]; then
  echo -ne \\f
fi
```

If you want a formfeed after 'lpr -l' to be optional you can misuse the '-i' switch to suppress the formfeed with the following trick (after all 'lpr -i -l' would usually not be implemented).

```
#!/bin/sh
cat
# use lpr -i -l to print raw without trailing formfeed
if [ "$1" != -c -o "$4" = -i0 ]; then
  # the ``echo -ne'' assumes that /bin/sh is really bash
  echo -ne \\f
fi
```

3.10 The Minimum /etc/printcap

This is a silly question but it is frequently asked. The answer is '`lp:sh`' (that's 6 bytes including the required linefeed character on the end). To use this `/etc/printcap` you must make `/dev/lp` a symlink to your printer and create your spool queue directory as `/usr/spool/lpd`. (You might think that if you wanted banner pages you could loose the '`:sh`' but the termcap syntax requires at least one capability per entry).

3.11 How to prevent the 'Staircase Effect'

Un*x terminates each line of a file with a linefeed but not a carriage return so taken literally a Un*x text file printed on an ASCII device will start each line below the end of the previous line. Some printers can be set to treat "linefeed" as "carriage return, linefeed", others can't. If yours can then do simply do that. If the printer cannot be fixed create a shell script filter that reads:

```
#!/bin/sh
if [ "$1" = -c ]; then
  cat
else
  sed -e s/$/^M/
fi
# the ``echo -ne'' assumes that /bin/sh is really bash
echo -ne \\f
```

Where `^M` is a carriage return character not a ^ followed by a `M`. To type `^M` in Emacs use the sequence `C-q C-m` and in vi use `C-v C-m`. Conventionally this script is called `/usr/lib/lpf`. If you have more than one such script a better idea is to keep them in a subdirectory, say `/usr/lib/lpd/`. The test of `$1` allows the insertion of carriage returns to be switched off by '`lpr -l`'.

Install this filter as the '`if`' filter by putting '`:if=/usr/lib/lpf:`' (or whatever) in your `/etc/printcap` entry for the printer.

Alternatively your printer may have an escape sequence that will set the way it handles linefeed characters. A simple filter that uses an '`echo -ne`' command to send this sequence may be appropriate.

```
#!/bin/sh
# Filter for HP printers to treat LF as CRLF
# the ``echo -ne'' assumes that /bin/sh is really bash
echo -ne \\033\&k2G
```

The list of capabilities can be as long as needed and the order is not significant. Each *capability* is denoted by a two character code. (The name "capability" comes form the file format's termcap heritage - parameter or attribute would be a more sensible terms.) [Note from Ross Biro: capabilities with 3 character names don't work properly which is why the serial port stuff in the old binaries failed.] Capabilities having string value and have a `=` delimiter between the capability name and the value while those having a numeric value use a `#` (actually they can use either a `#` or an `=`). Boolean "capabilities" are true if they appear in the list and false if they do not.

Special characters in a string value can be expressed using backslash-escaped sequences as in C; in addition, '`\E`' stands for `ESC`. '`^`' is also a kind of escape character; '`^`' followed by *CHAR* stands for the control-equivalent of *CHAR*. Thus, '`^a`' stands for the character control-a, just like '`\001`'. '`\`' and '`^`' themselves can be represented as '`\\`' and '`\^`' respectively. '`\:`' for '`:`' seems to work but the source code contains a warning that it can confuse the parser and '`\072`' is a better idea.

Example:

```
lp|bam|Epson FX-80:lp=/dev/lp1:sd=/usr/spool/lp1:sh:mx#0:\
        :df=/usr/local/lib/magic-filter/lp.df:\
        :if=/usr/local/lib/magic-filter/lp.if:
```

The printer's name is `lp` (this is the printer that `lpr` uses by default). It's also known as `bam` or `"Epson FX-80"`.

The printer is on `/dev/lp1` (aka AT-bus LPT1:). I don't want a burst page. I don't want a file length limit. Files queued by '`lpr -d`' are passed through `/usr/local/lib/magic-filter/lp.df` and those queued by `lpr` through `/usr/local/lib/magic-filter/lp.lf`.

See also the next section.

3.9 An `/etc/printcap` gotcha

Two `/etc/printcap` files can look identical and yet one works and the other doesn't.

See if '`lpc stat`' reports a printer called '` :`'. The last character on a continued line must be a backslash. If there are whitespace characters after the backslash then it doesn't register the next line as a continuation.

3.7 The Semantics of /etc/printcap

Given the similarity in appearance and name between `/etc/termcap` and `/etc/printcap` one could be forgiven for assuming that they contain analogous infomation. This is not the case. Whereas `/etc/termcap` contains informations about terminal **types** - (mostly escape seqences) printcap contains information about **specific** printers (like the directory that holds the spool queue, the device name of the printer and what room it's in). The information about a printer model's escape sequences and so on are held in the various *filters* which are programs called by `lpd` to drive the printer. `/etc/printcap` simply gives the locations of these filters. For details RTFM(printcap). [Alternatively the net-HOWTO has a summary of some of the more important fields.]

One last point - you should always specify 'suppress header' '`:sh:`' unless you have a **text** (not PostScript) printer and want banners. On a text printer they are usually a waste of time and paper. On a PostScript printer they usually stop your printer working. See Section 3.17 [Burst/banner pages], page 39.

3.8 The Syntax of /etc/printcap

Ideally RTFM(termcap) (yes, I said **termcap**) but since most people don't have TFM(termcap) here are the essentials.

Lines starting with '`#`' are comments (as you might have guessed).

For each printer usable from the `lpr` command on your system there is one logical line in the file. For the sake of readability each logical line may be spread over several physical lines by making the last character on all but the last physical line a backslash.

Each logical line has the following format:

```
name1|name2|name3:string_capability=string:\
        :numeric_capability#number:boolean_capability:
```

The leading spaces and colon on the second line are for readability only.

A printer can have as many names as you like but conventionally the final name is used as a longhand description of the printer. (Still people are free to say '`lpr -P "grotty teletype in room 213"`' if that's the description you've given.) One of the names of your default printer must be '`lp`'.

3.5 lpd not working

If 'ps ax' does not reveal a lpd then you daemon has died (or was never started) - usually a sign that it couldn't create its lockfile, or was unable to find /etc/services. (This will happen if you tried to start it before all your filesystems were mounted).

If lpr works only for root then you've probably got a permission problem.

If you cant even cat files to the printer then you may be using the wrong device name for the printer in /etc/printcap See Section 2.13 [Printer device names], page 14 or you may need to fiddle with tunelp. See Section 2.12 [hardware and drivers], page 13.

If you get "jobs queued, but cannot start daemon" or "lpc: connect: No such file or directory" while lpd **is** running then you are having trouble with the socket connection to lpd. "start" in the context of this error really means "wake". This problem has come and gone thoughout the history of Linux - I don't really understand this but it stems from an erroneous interaction between the networking stuff and "Unix domain" (non-network) sockets. Usually it has only shown up when the network is incorrectly configured. If you're not really on a network it is usually adequate just to have the following somewhere in your startup.

```
ifconfig lo localhost
route add localhost
```

You'll also need to have the /etc/hosts file. There's no need to run any daemons.

There is second and much more understandable way to produce this error - use a mixture of components from different releases of lpd that use different names for the Unix domain socket (new stuff uses /tmp/.printer, obsolete stuff /dev/printer). (For some time SLS was released this way).

At the time of writing I am quite unable to reproduce this error - I am using my debugged version of the net-2 lpd compiled with gcc-2.4.5 and libc-4.4.4 on kernel 0.99.14.

3.6 Where Do I Get A Printcap For Printer Model xxxxx?

This question is essentially meaningless so please don't ask it on usenet See Section 3.7 [The Semantics of /etc/printcap], page 32.

In older versions the group id was 'daemon' not 'lp'. There's also a named socket but `lpd` creates/deletes it as needed. Also you may find the uid is sometimes set to 'daemon' or 'lp' rather than 'root' but this is of no consequence except on setuid binaries.

These permissions are the most pedantic so don't be surprised if your system works with different permissions. On the other hand if lpd is not working for you and you have different permissions please try these before mailing us or posting to usenet. (This may sound obvious but you'd be amazed how much mail I've had from people with the permissions set wrong).

People tell me that `lpr` must be setuid(root) but I've not seen evidence that this is really the case as long as the file permissions on the spool queues are right. Still as far as I know `lpr` is designed to be secure when installed setuid(root). This allows an alternative (sloppy) approach: just make `lpr` and `lprm` setuid(root) then you can almost forget the file permissions on the spool queues!

You're free to choose different directories for the executables on your system (notably `lpc` is usually in `/etc` (or `/usr/sbin`) even though it has commands that are useful to non-root). Of course since you are free to do this it implies the person who made up your distribution was also free to do so, so be carefull to make sure you delete old components when you install new ones.

The master lpd lock file is fixed at compile time to be `/var/spool/lpd/lpd.lock` so you must have a `/var/spool/lpd directory` even if you choose not to keep your spool queue there. In the older binaries the lock file was `/usr/spool/lpd.lock` so this was not an issue.

My advise is to keep your primary spool queue in `/usr/spool/lpd` and make `/var` a symlink to `usr` or keep it in `/var/spool/lpd` and make `/usr/spool` a symlink to `../var/spool`. This gives the greatest compatibility with the pathnames that are compiled into the various distributed binaries. Note that having a separate `/var` filesystem avoids the problem of your `/usr` filesystem filling up as a result of stuff in spool queues.

The main configuration file is `/etc/printcap`. Network printing also uses `/etc/hosts.allow` and `/etc/hosts.lpd`.

By now everyone should have libraries and binaries that look for config files in `/etc`. If you chose to keep your configs somewhere else (`/conf` or `/usr/etc` for example) then `/etc` must contain a symlink to the real file. If you still have a system which looks for files in `/usr/etc` or `/etc/inet` your system is way out of date and you should upgrade.

Setting up lpd correctly is definitely worth the trouble if you are going to do any printing at all. Of course, if all you ever print is the occasional man page or instruction book, then you can probably get by without it (but even then it's nice).

Ghostscript 2.6.x also comes with a complicated script and utilities to generate printcap entries and manage multiple types of queues for a printer. I have not tested these, but I assume they can be made to work. If you are in a large-scale environment with varied printing needs, these may be worth a look.

3.4 Where Do The Files Go And What Should Their Permissions Be?

There is quite a bit of variation between the various releases but I'll try where possible to offer solutions that are universally applicable. Start `lpd` in your `/etc/rc` or `/etc/rc.local` (usually in `/etc/rc.local` after you start `syslogd` (if you use `syslogd`)).

Set the group fields of the file permissons/ownership as follows:

```
-rwxr-sr--   1 root     lp          37892 Nov 19 23:32 /etc/lpd
-rwxr-sr-x   1 root     lp          21508 Nov 19 23:32 /usr/bin/lpc
-rwsr-sr-x   1 root     lp          17412 Nov 19 23:32 /usr/bin/lpq
-rwxr-sr-x   1 root     lp          17412 Nov 19 23:32 /usr/bin/lpr
-rwxr-sr-x   1 root     lp          17412 Nov 19 23:32 /usr/bin/lprm
-rwxr-xr-x   1 root     lp           2816 May 10 13:37 /usr/bin/lptest

...and for each of the spool directories listed in the sd fields of
/etc/printcap...

/var/spool/lpd:
total 5
drwxrwxr-x   2 root     lp           1024 May 18 23:00 .
drwxr-xr-x  11 root     root         1024 Feb 19 20:56 ..
-rw-rw-r--   1 root     lp              4 May 18 23:00 .seq
-rw-rw-r--   1 root     lp             18 May 18 23:00 lock
-rw-rw-r--   1 root     lp             25 May 18 23:00 status
```

Note these 3 files are created by lpr and lpd so if you've never run these they will be absent. With some versions of the lpd package you had to touch them into being or they would be created with the wrong permissions but this bug seems to have been fixed.

machine may have been downed by the system administrator. See Section 3.23 [lpr over a network], page 42.

3.2 What lpr and lpd do

Most Un*x systems use `lpd` (or the System V variant `lp`), the line printer daemon, and friends to spool print jobs and run them through all needed filters. While line printers are certainly on their way out, spooling print jobs and running them through filters is definitely a convenient thing. Thus lpr. (subliminal note: **PLEASE** find and read the `lpr`, `lpd`, `printcap`, `lpc`, `lpq`, and `lprm` man pages. They are in the source dist of the net-2, if you haven't got them.)

3.3 Getting hold of lpd

Lpd and family are available in several places. The new Linux net-2 package contains a working BSD-style lpd (although it has a couple of bugs and Makefile installs the binaries with the wrong permissions). I now use this one with my stock 0.99pl14 kernel after following the directions in Matt Welsh's Net-2 HOWTO to get my NET-2 going (you only need loopback). Most people (and this document) consider this to be **the** Linux lpd package. A slightly debugged version incorporating patches from earlier releases of this HOWTO is also available in:
`tsx-11.mit.edu:/pub/linux/packages/net/new-net-2/lpd-590p?.tar.gz`

Some of the prepackaged Linux distributions have a very poor record with respect to having a working lpd packages so don't be shy of getting a new one if you seem to be having trouble.

Fans of the SysV lp package should consider getting plp (however this HOWTO contains no information on that package)

An lpd binary package which can be coaxed into working with the old net things (pre 0.99pl10 kernel) may still be around but you'll probably find it a bit shakey and you really should update your kernel. Note a subtle difference between the two versions: the old one placed lock files and such in `/var/spool`, while the net-2 version requires the directory `/var/spool/lpd` to exist for its lock file.

You will need to edit `/etc/printcap` to configure lpd to use your printer. Setting up lpd to accept PostScript and print it is not very difficult; simply make a shell script filter. See Section 3.24 [Writing lpd filters], page 43

Answer: You have not created the spool directory specified in the printcap entry or have misnamed it. An alternative (though much less likely) answer is that you have too little disk space left.

Problem: You get a message saying "lpr: Printer queue is disabled".

Answer: As root, use '`lpc enable` <*printername*>' to enable the printer. Note that as root, you can submit jobs even to a disabled printer.

Problem: You submit a print job, there are no error messages, but nothing comes out on the physical printer.

Answer: There could be many reasons. Make sure the physical printer is switched on, selected, and physically connected to the device specified in the `/etc/printcap` file. Use the `lpq` command to see whether the entry is in the queue. If it is, then the device may be busy, the printer may be down, or there may be an error on the printer. Check the error log specified in the printcap entry for clues. You can use the '`lpc status`' command to check whether the printer is down and '`lpc up` <*printername*>' to bring it back up if it is (you need to be root to do this).

If after checking as suggested your print jobs still do not come out, double check that any input filter you have specified is present in the correct directory and has the correct permissions. If you are running syslogd, you can look in your logs for messages from lpd. If you see log entries saying "cannot execv <name of input filter>", then this is almost certainly the problem.

Another possibility is that your printer is a PostScript printer and you are not sending PostScript to it. Most PostScript printers will ignore non-PostScript data. You may need to install an appropriate text-to-PostScript input filter.

Lastly (and you'll feel really silly if this is the cause!) check that your input filter actually generates output and that the output device is not `/dev/null`.

Problem: When remote printing, your jobs go into the remote queue but never get physically printed.

Answer: If you can, look at the entry for the remote printer in the `/etc/printcap` file on the remote machine. It may be restricting your access. The '`rg`' field restricts access to members of a specific group which you may not be in, and the '`rs`' field prevents access from users without accounts on the remote machine, which you may not have. Alternatively the printer on the remote

To add more printers, just repeat the above process with different printer names. Remember you can have multiple printcap entries all using the same physical device. This lets you treat the same device differently, depending on what you call it when you submit a print job to it.

It is possible to "reuse" a printcap entry. If you specify your own machine as the remote host and another printer in your printcap file as the remote printer, you can effectively redirect print data from one printer to another. Remember if you use this technique that all the data will be processed by all input filters in the chain and spooled for each printer it goes through.

Although you can specify as many aliases for a printer as you like, it seems that for maximum usefulness the first two should be the same and should be the "real" printer name. Many programs will only ever use one of these two aliases. The `lpc` command will only report on the first alias, though `lpc`, `lpr`, `lprm` and `lpq` all understand any alias.

Rather than specify a maximum spool file size, you may want instead to prevent spool files expanding to fill your disk, even temporarily. To do so, put a file called minfree in each spool directory, specifying the amount of disk space that must remain for spooling data to be accepted. This file is a simple text file, containing the number of blocks to be left free. Usually this file is a link to a file in the main spool directory, as it is rare for different printers to have different minimums.

3.1.23 Troubleshooting

Problem: You get a message saying "lpd: connect: No such file or directory"

Answer: The printer daemon `/etc/lpd` is not running. You may have forgotten to add it to your `/etc/rc.local` file. Alternatively you did add it, but haven't booted since. Add it and reboot, OR just run /etc/lpd. Remember you have to be root to do this. See Section 3.5 [lpd not working], page 31.

Problem: You get a message saying "Job queued, but cannot start daemon".

Answer: This often appears right after the "lpd: connect" message. Same problem.

Problem: You get a message saying "lpd: cannot create <spooldir>/.seq".

11. Do a test print again - this one should come out on your physical printer!

```
ls -l | lpr -Pmyprinter
```

3.1.21 More On Remote Printing

In order for any other machines to print using your printers, their names will have to be registered in either the file `/etc/hosts.equiv` or `/etc/hosts.lpd`. These are simple text files, one host name per line.

For preference, add hosts to `/etc/hosts.lpd`. `/etc/hosts.equiv` is used to give far wider access rights, and should be avoided wherever possible.

You can restrict remote users either by group name (specify the groups permitted using one or more rg fields in `/etc/printcap` - '`:rg=admin:`' will restrict access to a printer to those users belonging to the group admin. You can also restrict access to those users with accounts on your system, by specifying the boolean flag '`:rs:`' in your `/etc/printcap`.

3.1.22 The Fiddly Bits

If all the above worked, you will now have two printers defined in `/etc/printcap` - one called testlp, which appends output to `/tmp/testlp.out`, and one called myprinter which sends unmodified output to the physical printer attached to `/dev/lp0`. Both share the same spool directory. As an exercise, you might like to set up a separate spool directory for the testlp printer.

If your printer is a PostScript printer, it may not be able to handle plain text. If this is the case, you'll need to set up a filter to convert plain text to PostScript. An excellent freeware program called nenscript is available which does just this. See Section 2.3 [PostScript], page 3. If you don't set up such a filter, you will have to make sure by other means that the printer is sent only PostScript.

You may like to add a line to your login script - or even to the default user login script - which sets up a `PRINTER` environment variable. Under bash, a suitable line would be '`export PRINTER=myprinter`'. This will prevent people having to specify '`-Pmyprinter`' every time they submit a print job.

3.1.20 Putting It All Together

Putting all the above bits together, here is a step by step guide to setting up a single printer on `/dev/lp0`. You can then extend this to other printers. You have to be root do do all this, by the way.

1. Check the permissions and locations of `lpr`, `lprm`, `lpc`, `lpq` and `lpd` See Section 3.1.4 [The important programs], page 17.
2. Create the spool directory for your printer, which we will call '`myprinter`' for now. Make sure both are owned by root, group daemon, and are user and group writeable, readonly for others (-rwxrwxr-x).
   ```
   mkdir /usr/spool/lpd /usr/spool/lpd/myprinter
   chown root.daemon /usr/spool/lpd /usr/spool/lpd/myprinter
   chmod ug=rwx,o=rx /usr/spool/lpd /usr/spool/lpd/myprinter
   ```
3. In the directory `/usr/spool/lpd/myprinter`, create the necessary files and give them the correct permissions and owner:
   ```
   cd /usr/spool/lpd/myprinter
   touch .seq errs status lock
   chown root.daemon .seq errs status lock
   chmod ug=rw,o=r .seq errs status lock
   ```
4. Create the shell script input_filter in the directory `/usr/spool/lpd/myprinter`. Use the input filter given above. Make sure that the file is owned by root, group daemon, and is executable by anyone.
   ```
   cd /usr/spool/lpd/myprinter
   chmod ug=rwx,o=rx input_filter
   ```
5. Create the file `/etc/printcap` if it doesn't already exist. Remove all entries in it and add the test printcap entry given above. Make sure the file is owned by root, and readonly to everyone else (-rw-r–r–).
6. Edit the file `rc.local`. Add the line '`/etc/lpd`' to the end. This will run the printer daemon each time the system boots. It is not necessary to boot now though - just run it by hand:
   ```
   lpd
   ```
7. Do a test print:
   ```
   ls -l | lpr -Pmyprinter
   ```
8. Look in `/tmp` for a file called `testlp.out` - it should contain your directory listing.
9. Edit `/etc/printcap`. Copy the myprinter entry, so you have two identical entries.

   in the **first** entry, change both occurrences of '`myprinter`' to '`testlp`' in the first line only

   in the **second** entry, change `/dev/null` to your real print device, eg., `/dev/lp0`

   in the **second** entry, remove the '`if`' line completely
10. Either reboot the system or kill the printer daemon and restart it. This is because the printer daemon only looks at the /etc/printcap file when it first starts up.

data, **not** the amount of data sent to the physical printer. If a user tries to exceed this limit the file is tuncated. The user will see a message saying "lpr: <filename>: copy file is too large".

For text physical printers, this is useful if you have users or programs that may deliberately or accidentally create excessively large output, but in most cases is not really very applicable.

For PostScript physical printers, the limit is not useful at all, because a very small amount of spooled PostScript data can generate a large number of output pages.

3.1.19 A Test Printcap Entry

The following shell script is a very simple input filter - it simply concatenates its input onto the end of a file in `/tmp` after an appropriate banner. We specify this filter in our printcap entry and specify `/dev/null` as the print device. The print device will never actually be used, but we have to set it to something because otherwise the printer daemon will complain.

```
#!/bin/sh
# This file should be placed in the printer's spool directory and
# named input_filter. It should be owned by root, group daemon, and be
# world executable (-rwxr-xr-x).
echo ------------------------------------------------ >> /tmp/testlp.out
date                                                  >> /tmp/testlp.out
echo ------------------------------------------------ >> /tmp/testlp.out
cat                                                   >> /tmp/testlp.out
```

Here's the printcap entry. Notice the (reasonably) readable format and the use of continuation characters on all but the last line:

```
myprinter|myprinter: \
:lp=/dev/null: \
:sd=/usr/spool/lpd/myprinter: \
:lf=/usr/spool/lpd/myprinter/errs: \
:if=/usr/spool/lpd/myprinter/input_filter: \
:mx#0: \
:sh: \
:sf:
```

3.1.15 More On The if Field

Input filters are programs which take print data on their standard input and generate output on their standard output. A typical use of an input filter is to detect plain text and convert it into PostScript. That is, raw text is its input, PostScript is its output. See Section 3.24 [Writing lpd filters], page 43.

When you specify an input filter, the printer daemon does **not** send the spooled print data to the specified device. Instead, it runs the input filter with the spooled data as standard input and the print device as standard output. (For another use for input filters see Section 3.1.19 [A test printcap entry], page 23).

3.1.16 More On The `rm` and `rp` Fields

Sending your print data to a printer attached to another machine is as simple as specifying the remote machine '`rm`' and the remote printer '`rp`', and making sure that the print device field '`lp`' is empty. Note that data will still be spooled locally before being transferred to the remote machine, and any input filters you specify will be run also.

3.1.17 More On The `sh` and `sf` Fields

Unless you have a lot of different people using your printer, you will most likely not be interested in banner pages.

Suppressing form feeds is most useful if your printer is typically used for output from wordprocessing packages. Most WP packages create complete pages of data, so if the printer daemon is adding a form feed to the end of each job, you will get a blank page after each job. If the printer is usually used for program or directory listings, however, having that form feed ensures that the final page is completely ejected, so each listing starts at the top of a new page.

3.1.18 More On The `mx` Field

This field allows you to limit the size of the print data to be spooled. The number you specify is in BUFSIZE blocks (1K under Linux). If you specify zero, the limit is removed, allowing print jobs to be limited only by available disk space. Note that the limit is on the size of the spooled

3.1.12 Fields in `/etc/printcap`

There are too many fields to describe here in full, so I'll just describe the most important ones. All fields in `/etc/printcap` (except for the names of the printer) are enclosed between a pair of colons and are denoted by a two-letter code. The two-letter code is followed by a value that depends on the type of field. There are three types of field - string, boolean and numeric See Section 3.8 [The Syntax of /etc/printcap], page 32.

The following fields are the most common and most important ones:

```
lp      string          specify the device to print to, eg /dev/lp0
sd      string          specify the name of the spool directory for
                        this printer
lf      string          specify the file to which errors on this
                        printer are to be logged
if      string          specify the input filter name
rm      string          specify the name of a remote printing host
rp      string          specify the name of a remote printer
sh      boolean         specify this to suppress headers (banner pages)
sf      boolean         specify this to suppress end-of-job form feeds
mx      numeric         specify the maximum allowable print job size
                        (in blocks)
```

3.1.13 More on The '`lp`' Field

If you specify `/dev/null` as the print device, all other processing will be performed correctly, but the final data will go to the bit bucket. This is rarely useful except for test printer configurations or with weird printers See Section 3.16 [Printers not in /dev], page 39. When you are setting up a remote printer (that is, you have specified '`rm`' and '`rp`' fields), you should specify '`:lp=:`'.

Don't leave the field empty unless you are using a remote printer. The printer daemon will complain if you don't specify a print device.

3.1.14 More On The lf Field

Whatever file you specify should already exist, or logging will not occur.

3.1.11 More About /etc/printcap

The file `/etc/printcap` is a text file, which may be edited with your favourite editor. It should be owned by root and have the permissions "-rw-r–r–".

The contents of `/etc/printcap` are typically very cryptic-looking, but once you know how it works they are much easier to understand. The problem is compounded by the fact that in some braindead distributions there is no man page for printcap, and the fact that most printcaps are created either by programs or by people with no thought for readability. For your own sanity, I recommend making the layout of your printcap file as logical and readable as possible, with lots of comments. And get the a man page from the lpd sources, if you don't already have it.

One printcap entry describes one printer. Essentially a printcap entry provides a logical name for a physical device, then describes how data to be sent to that device is to be handled. For example, a printcap entry will define what physical device is to be used, what spool directory data for that device should be stored in, what preprocessing should be performed on the data, where errors on the physical device should be logged and so forth. You can limit the amount of data which may be sent in a single job, or limit access to a printer to certain classes of user.

It is perfectly OK to have multiple printcap entries defining several different ways to handle data destined for the same physical printer. For example, a physical printer may support both PostScript and HP Laserjet data formats, depending on some setup sequence being sent to the physical printer before each job. It would make sense to define two printers, one of which preprocesses the data by prepending the HP LaserJet sequence while the other prepends the PostScript sequence. Programs which generate HP data would send it to the HP printer, while programs generating PostScript would print to the PostScript printer.

Programs which change the data before it is sent to the physical printer are called “filters”. It is possible for a filter to send no data at all to a physical printer.

See Section 3.8 [The Syntax of /etc/printcap], page 32

```
# Sample printcap entry with two aliases
myprinter|laserwriter:\
# lp is the device to print to - here the first parallel printer.
:lp=/dev/lp0: \
# sd means 'spool directory' - where print data is collected
:sd=/usr/spool/lpd/myprinter:
```

3.1.9 The Important Directories

There is really only one important directory - the spool area where data to be printed is accumulated before `/etc/lpd` prints it. However, typically a system will be set up with multiple spool directories, one for each printer. This makes printer management easier. My system is set up to use `/usr/spool/lpd` as the main spool area, with each separate printer having a directory under that with the same name as the printer. Thus I have a printer called '`ps_nff`' which has `/usr/spool/lpd/ps_nff` as its spool directory and so on.

The spool directories should belong to the daemon group and be user and group read/writable, and world -readable. That is, after creating the directory make sure it has permissions "-rwxrwxr-x" (0775). For the directory myprinter, the appropriate command would be:

```
chmod ug=rwx,o=rx myprinter
chgrp daemon myprinter
```

(The locations, ownerships and permissions given here are a simplification and may be wrong for your system see Section 3.4 [lpd files and permissions], page 29)

3.1.10 The Important Files

Apart from the programs discussed above, each spool directory should contain four files - `.seq`, `errs`. `lock` and `status`. These files should have the permissions "-rw-rw-r–". The `.seq` file contains the job number counter for `lpr` to assign a and the `status` file contains the message to be reported by '`lpc stat`'. The `lock` file is used by `lpd` to prevent itself trying to print two jobs to the same printer at once, and the `errs` file is a log of printer failures.

The file `errs` is not required and can actually be called whatever you like - the name is specified in `/etc/printcap`, but the file must exist for `lpd` to be able to log to it, so it is usually created manually when setting up the spool area. More on this later.

One very important file is the file `/etc/printcap`, which is described in detail in the following sections.

3.1.6 The lpq Command

The lpq command shows you the contents of the spool directory for a given printer. One important piece of information displayed by lpq is the job id, which identifies a particular job. This number must be specified if you wish to cancel a pending job.

lpq also shows a rank for each job in the queue - "active" means the file is actually printing (or at least that lpd is trying to print it). Otherwise a number shows you where in the queue the job is.

3.1.7 The lprm Command

The lprm command removes a job from the queue - that is, it removes unprinted files from the spool directory. You can either specify a job id (obtained by using the lpq command) or specify '-' as the job id, in which case all jobs belonging to you will be cancelled. If you do this as root, all jobs for the printer will be cancelled. If you are root and want to remove all the jobs belonging to a specific user, specify the user's name.

3.1.8 The lpc Command

The lpc command lets you check the status of printers and control some aspects of their use. In particular it lets you start and stop despooling on printers, lets you enable or disable printers and lets you rearrange the order of jobs in a print queue. The following commands disable printing on myprinter, enable the spool queue on yourprinter and move job number 37 to the top of the queue:

```
lpc down myprinter
lpc enable yourprinter
lpc topq 37
```

If invoked without any command arguments, lpc will be interactive, prompting you for actions to take. Read the man page for complete instructions. Bear in mind that some lpc functions are restricted to root.

In the discussions that follow, the term "printer" will be used to mean a printer as specified in `/etc/printcap`. The term "physical printer" will be used to mean the thing that actually puts characters on paper. It is possible to have multiple entries in `/etc/printcap` which all describe one physical printer, but do so in different ways. If this is not clear to you, read the section on `/etc/printcap`.

3.1.4 The important programs

There are five programs which comprise the Unix print system. They should be in the locations shown, should all be owned by root and belong to the group daemon and have the permissions shown here:

```
-rwsr-sr-x /usr/bin/lpr
-rwsr-sr-x /usr/bin/lpq
-rwsr-sr-x /usr/bin/lpc
-rwsr-sr-x /usr/bin/lprm
-rwxr-s--- /etc/lpd
```

The first four are used to submit, cancel and inspect print jobs. `/etc/lpd` is the printer daemon.

(The locations, ownerships and permissions given here are a simplification and may be wrong for your system see Section 3.4 [lpd files and permissions], page 29)

There are man pages for all these commands, which you should consult for more information. The important points are that by default `lpr`, `lprm`, `lpc` and `lpq` will operate on a printer called '`lp`'. If you define an environment variable called `PRINTER`, the name thus defined will be used instead. Both these may be overridden by specifying the printer name to use on the command line thus:

lpc -P*myprinter*

3.1.5 The `lpr` Command

The `lpr` command submits a job to the printer, or "queues a print job". What actually happens is that the file you specify is copied to the spool directory (see above), where it will be found by `lpd`, which then takes care of moving the data to the physical printer. If you don't specify a file, `lpr` uses standard input.

It also assumes that you have your Linux system set up and running correctly. In particular, if you are going to use remote printing your networking subsystems must be installed and operating correctly.

Check out the man pages on the commands `chmod` and `chown` for more information.

3.1.3 How Printing Works Under Linux

The simplest way to print under Unix (and thus under Linux) is to send the print data directly to the printer device. This command will send a directory listing to the first parallel printer (LPT1: in DOS terms):

```
ls > /dev/lp0
```

This method does not take advantage of the multitasking capabilities of Linux, because the time taken for this command to finish will be however long it takes the printer to actually physically print the data. On a slow printer, or a printer which is deselected or disconnected, this could be a long time.

A better method is to spool the data. That is, to collect the print data into a file, then start up a background process to send the data to the printer.

This is essentially how Linux works. For each printer, a spool area is defined. Data for the printer is collected in the spool area, one file per print job. A background process (called the printer daemon) constantly scans the spool areas for new files to print. When one appears, the data is sent to the appropriate printer or **despooled**. When more than one file is waiting to be printed, they will be printed in the order they were completed - first in, first out. Thus the spool area is effectively a queue, and the waiting jobs are often referred to as being “in the print queue”, or “queued”.

In the case of remote printing, the data is first spooled locally as for any other print job, but the background process is told to send the data to a particular printer on a particular remote machine.

The necessary information that the printer daemon needs to do its job - the physical device to use, the spool area to look in, the remote machine and printer for remote printing and so on - is all stored in a file called `/etc/printcap`. The details of this file are discussed below.

3 LPR

This chapter used to the the `lpd-FAQ` then it became part of the `Linux-FAQ` and now it is part of the Printing-HOWTO. Who knows where it will go next?

3.1 Setting up print services

This section was originally writen Karl Auer <Karl.Auer@anu.oedu.au> 1/11/93 and has now been included here because it was felt that the Printing-HOWTO, being written in the style of a reference manual, was not doing its job properly, since it did not describe in a straightforward way what you need to do to set up print services on your Linux machine.

So far it has been included with little modification and may replicate or even contradict information elsewhere in the HOWTO. The overview it gives should nevertheless be very useful to those new to bsd print spooling.

3.1.1 Remote Printing Vs. Local Printing

Remote printing is allowing people to send print jobs to your computer from another computer. This will be needed if, for example, you are running as a server in a network, or if a printer attached to your machine is to be accessible from other Unix hosts.

Local printing is allowing users on your machine to send print jobs to a printer attached to your machine.

There is a third combination too - your own use of remote printing on other Unix machines. That is, where you wish to print on a printer that is not attached to your own computer.

3.1.2 What You Need

This document assumes you know how to edit a text file under Linux, and that you have a basic understanding of file ownership and permissions.

already contain the correct i/o base address. Many bios boot screens have this information if you need it.

DOS uses a polling driver so if you find that your printer works in DOS but not in Linux then you should try the polling driver.

`tunelp` is available from the printing mail server or:
`sunsite.unc.edu:/pub/Linux/system/Printing/tunelp-1.0.tar.z`

2.13 Printer device names

On an XT bus system `LPT1:` becomes `/dev/lp0` (major=6, minor=0), on an AT `LPT1:` becomes `/dev/lp1` (major=6, minor=1). To be more precise:

```
Name Major Minor I/O address
 lp0   6     0       0x3bc
 lp1   6     1       0x378
 lp2   6     2       0x278
```

For a serial printer use the `/dev/ttyS?` (or `/devttys?`, if you follow that scheme) device, not the `/dev/cua?` device. The `/dev/ttyS?` devices have major 4; the `/dev/cua?` devices have major 5.

There are also separate programs which read and format info from a texinfo file. These are available in:
`prep.ai.mit.edu:/pub/gnu/`

2.11 Printing though a terminal

Many terminals and terminal emulators support the connection of a printer. The computer the terminal is conected to may then use a standard set of escape sequences to operate this printer. DEC VT's and most other ANSI terminals should offer this capability.

A simple shell script will enable you to print a text file using the printer connected to your terminal:

```
#!/bin/csh -f
# Transparent printing on a vt100-compatible terminal.
echo -n \[5i ; cat $* ; echo -n \[4i
```

If this script is called `vtprint`, then '`vtprint [`*filename* `[`*filename*`]]`' will print file(s) (or stdin if no file were specified) on your printer.

2.12 hardware and drivers

There are two ways the kernel driver may be used to run the parallel ports. One, the original, is the polling driver. The other is the interrupt driver. In principle, the interrupt driver only deals with the port when it gets an interrupt and should therefore be more efficient. In practice, people have found that it depends on the machine. It probably doesn't make too much difference in most situations.

For the polling driver, you may adjust its polling frequency with the program tunelp without kernel twiddling. The actual driver is in the kernel source file `lp.c`.

To choose the interrupt driver rather than the polled, use the program tunelp to set it. (tunelp is available on sunsite, or from the printing mail server.) Just put the appropriate line in `/etc/rc.local`

Seven is the usual "LPT1:" IRQ, 5 is usual for "LPT2:" for an AT class machine. Note that if your printer is on lp0, the interrupt driver will probably not work. The first parameter should

```
sunsite.unc.edu:/pub/Linux/apps/tex/eps-061.tar.gz
ftp.ctd.comsat.com:/pub/
```

Note that it is still coming out with new versions fairly often, so there may be a newer version than 061.

2.9.3 dvilj

For LaserJet owners, there is a separate program that will take dvi and convert it directly into your printer's language (PCL). It is called dvilj2p or dvilj, and if not on tsx or sunsite, is certainly available on ftp.uu.net.

(Description by Nils Rennebarth)

> Its a nice driver, but a little out of fashion in the sense that configuration (especially of font-paths) font-paths is not very flexible and that it doesn't support virtual fonts (at least the version 0.51 not). The biggest advantage over the dvips/ghostscript combo is that it uses downloadable fonts which:
>
> > reduces data transmission to the printer drastically, which makes the printer usable even on a serial line.
> >
> > reduces printer-memory-requirements drastically. A standard Laserjet with 512k memory is able to print almost every TeX-document.
>
> It has support for double side printing and most options you expect a driver to have. It compiles cleanly and worked flawlessly on our diverse hardware here.

2.10 texinfo

This is the native documentation format of the GNU project. Emacs can be coerced into producing an info file from TeXinfo, and TeX can produce nice printed documantation from the same file. It is a bit of a stretch for both systems, but it works. It is really just TeX source which expects the macro file `texinfo.tex` to be installed on your system. Just do '`tex` *filename*' twice (for index generation purposes), and you end up with a plain dvi file, to print or preview at your leisure.

In Emacs, you can also do `M-x texinfo-format-buffer` to convert the texinfo file into an info file viewable with Emacs `M-x info` or an info viewer of your choice.

2.9 TEX/dvi

[La]TEX is the text formatting package used by many in the academic world and elsewhere. TEX works much like any other compiler – source code is run through the program `tex` to become a `.dvi` file (analogous to an .o object file in C) which can be further manipulated to produce printed output (a "binary"). This manipulation of a dvi (DeVice Independant) file usually takes a little bit of doing. It is well worth it; TEX's output is of professional quality.

If all you are given is a file with a `.tex` ending, try either '`tex file.tex`' or '`latex file.tex`'. One of these is bound to work. Then you have a dvi. (You may have to run it twice for indexing)

For those in the real world who cannot afford a dvi understanding printer, it is usually necessary to convert the dvi something the printer understands. These programs may be run manualy but are often built into lpd filters. See Section 3.24 [Writing lpd filters], page 43.

2.9.1 dvips

`dvips` converts dvi into PostScript that you can pipe into ghostscript or send to a PostScript printer. Most installations come with functioning installations of both TEX and `dvips`. Typing '`dvips -f1 file.dvi | lpr`' will do it. `dvips` responds to either command line arguments or a file `/usr/TeX/lib/tex/ps/config.ps` (in the usual TEX layout, at least) in which you can arrange to have dvips automatically send its output to `lpr`. Thus '`dvips file.dvi`' will do everything that needs to be done.

Note that some `.dvi`'s may include PostScript graphics in the `dvips` stage rather than the TEX stage of the game; if they are not found, you will get a hole instead of a picture. This follows naturally from the object file analogy above. Usually, pre-made documentation in this form has a makefile or script to do everything for you. The LILO documentation is an example of this.

`dvips` has several interesting options; for example, '`dvips -r1 file.dvi`' will print it out backwards. We deskjet users love this one.

2.9.2 eps

Eps is a program which converts dvi files directly into the standard Epson printer language; thus it is a dvi driver for epsons. I beleive it supports MakeTeXPK, the automatic font rendering system used by many dvi drivers, and is available as:

at this. It is available at the same place mpage is. Not that if you have a deskjet, many n-up programs will run into trouble by trying to print in that last half-inch.

2.7.3 enscript

Enscript is a program which does basically the same thing as a2ps. I do not know where to get it. It comes with most commercial Un*ces.

A clone version of enscript is called nenscript, available on `sunsite.unc.edu` as:
`/pub/Linux/system/Printing/nenscript-1.13++.bin.tar.z`
`/pub/Linux/system/Printing/nenscript-1.13++.tar.z`

2.7.4 gslp

Gslp is one of the uilities which comes with ghostscript 2.6.x and purports to do the same ascii ⇒ ps conversion as enscript and a2ps. I have not used it, but the docs say that

`gs -q -sDEVICE=`*?????* `-dNOPAUSE -- gslp.ps` *text.file [options]*

should do the trick. (`gslp.ps` is the actual program, which is written in PostScript. Here it is run with the argument *text.file*. PostScript is in many respects a programming language more than a mere printer command language.) Further documentation is in the file '`gslp.ps`'. There is a script file which does the above for you.

2.8 PostScript utilities

Those of you who deal with large amounts of PostScript may wish for more utility programs. There are probably millions of little programs which do things to your PostScript. A representative package of them may be found in:
`achilles.doc.ic.ac.uk:/tex/inter/psutils/`
`ftp.cs.psu.edu:/pub/src/psutil.tar.gz`

These handle page selection, double-sided printing, booklet creation, etc. Most large ftp sites (eg, `wuarchive.wustl.edu`, `ftp.uu.uunet`) will have many such packages to choose from.

If you have the nroff source to the page (the finding of which I highly recommend) you can say '`man -t foobar | lpr`' and your man program will (hopefully) format the man page using groff into PostScript, which will then be sent to your lpd and on to the printer. This form of man page output looks MUCH better than the plain ASCII version. Unfortunately, this depends very much on which man program and what supporting software you have installed. If your man doesn't do this, you might try the perl version of man, available near:
`sunsite.unc.edu:/pub/Linux/system/Manual-pagers/`

It is written entirely in perl, and is thus easily customizable (perl being an interpreted language reminiscent of C and sh).

You can also find the nroff source file in the man directories (most versions of man have an option to just spit out the filename) and do it yourself, into any format supported by groff.

```
groff -mandoc -Ttype foobar.1 | lpr
```

Where *type* is one of `ascii`, `dvi`, `ps`, `X100`, `X75`, `latin8`.

2.7 Printing text via PostScript

There are a number of utilities to allow text to be printed to a PostScript device.

2.7.1 mpage

The package mpage converts plain text into PostScript and/or prints more than one page onto each peice of paper given PostScript or text. It is available at (or at least near):
`wuarchive.wustl.edu:/pub/mirrors/unix-c/PostScript/mpage.tar-z`

Note that wuarchive uses the '`-z`' suffix to mean '`.Z`', ie, `compress`, not `gzip` or `freeze`. '`man -t foobar | mpage`' will send a 2-up (depending on the environment variable MPAGE) version of the man page to lpr and its PostScript interpreter. This saves paper and speeds up printing.

2.7.2 a2ps

A2ps will take ASCII and turn it into a nice PostScript document with headers and footers and page numbers, printed two pages on one (or otherwise, if you say so). A2ps does a very nice job

already; this is yet another reason to get the source. Some fax sofware on commercial platforms can use the display PostScript renderer to render the fax image; this is not yet an option under Linux.

The GNU program netfax supports Class 2 fax modems only. It builds nearly out of the box under linux. I have patches for version 3.2.1; you can get them from the printing mail server as '`netfaxpatch`'. It runs a server and accepts jobs into a queue from the local machine and the net. Netfax is in use at MIT. It accepts PostScript, dvi, and ascii, and can be configured to work as an email gate. Documentation in the package is limited to compile notes and man pages for the individual programs; actual installation tips are few and far between.
`prep.ai.mit.edu:/pub/gnu/fax-3.2.1.tar.gz`

FlexFax is available from:
`sgi.com(192.48.153.1):/sgi/fax/`*?????*`.src.tar.Z`

It is written in C++, and thus requires g++ to build. FlexFax supports class 1 and 2 modems, uses ghostview (or a display PostScript server) for rendering, and generally is more complete, or at least general than the somewhat MIT-specific netfax. It also can run in place of getty and either receive a fax or spawn a login as appropriate for the type of call. It includes precise instructions on configuring smail3, sendmail, and other mail agents to send mail for so-and-so@12345678.fax to the fax at 12345678. This is the package I recommend if you have real faxing needs, as it is properly documented and full-featured. Version 2.x of FlexFax is in real release now and fully supports Linux - just type '`./configure`' and '`make`'.

mgetty+sendfax is a Linux and SCO-specific getty for faxmodems and a simple sendfax program. This package is available as `sunsite.unc.edu:/pub/Linux/system/Serial/mgetty+sendfax-0.16.tar.gz`. This is a good package for those who need to send a fax every now and then.

Finally, `efax` deserves mention. Class 2 only. no net, no mail, just send a tiff. Efax is available as:
`sunsite.unc.edu:/pub/Linux/apps/comm/efax05.tar.gz`

2.6 *roff, man pages

Man pages can sometimes be printed straight from the cat pages (in lieu of the normal nroff source pages) as though they were a normal text file (which they mostly are). However, many printers do not like the control characters for highlighting and what not that is imbedded in these. A filter for this purpose comes with the net-2 lpd package.

A full fontmap file for the ususal set of fonts included with the Adobe Type Manager is available as '`fontmap.atm`' from the printing mailserver.

TimesNewRomanPS and ArialMT fonts are interchangable with Times Roman and Helvetica, so you can alias them this way if this is what you have.

Adobe Type 1 fonts may be found on the net in several places:

`ftp.cica.indiana.edu:/pub/pc/win3/fonts`
`archive.umich.edu:/msdos/mswindows/fonts`

I have not looked in these places, but this information is lifted straight from the comp.fonts FAQ (which you should read if fonts are a thing for you. You can get a copy of this from `rtfm.mit.edu`).

Conversion between various font types is tricky. Ghostscript comes with the tools needed to take a bitmap (hopefully large) and make a scalable ps font. Groff comes with the tools to allow use of tfm/mf (TeX) and pfb (Type 1) fonts in *roff documents. X11R5 includes several font utilities and Type 1 rendering code contributed by IBM. I have used none of the above, but they all come with manpages, so read them. Other font conversion utilities are listed in the comp.fonts FAQ. Also look into the package fontutils on `prep.ai.mit.edu:/pub/gnu/`.

2.5 faxing

If you have a fax modem, you can arrage to fax things out (and in), including PostScript, dvi, ascii, etc... Arranging for e-mail to fax itself somewhere is also straightforward.

Fax modems support one of of two cammand sets: Class 1 or 2. Class one modems are have less of what goes on in a fax supported in firmware (thus the software has to do more. Supporting such timing-critical things under a preemptive multitasking environment like Linux is tricky at best). The class 1 standard is EIA 578. Class 2 modems tend to be more expensive and comply with the standard EIA 592. Mention of your fax modem's level of support should be in its manuals. Do not confuse class and group. You will, of course, want a group III fax modem.

Fax software which runs under Linux must convert input in whatever format into a Group III compatible image format for transmission. As usual, Ghostscript does the job. The device tiffg3 generates standard g3/tiff encoded fax messages. You must compile this device in if it is not there

A patch which uses the Linux svgalib, and hence does not require X for previewing is available. See Section 4.6 [non-X previewing], page 53.

A minimal binary installation of ghostscript and several other packages needed for printing the Linux documentation is available as:
`sunsite.unc.edu:/pub/Linux/apps/tex/texmin/texmin-0.1.tar.z`.

Note that this does not contain any PostScript fonts. (Nor do you need them to print dvi (aka [La]TEX).)

The main Ghostscript documentation is contained in the file `use.doc`, either in the source directory, or `lib/ghostscript/doc/use.doc` if you haven't the sources.

To print ps, first determine your driver name with '`gs -help`' which lists installed drivers. If the device you need is not listed, you must compile gs yourself from the source distribution (do not panic. Do follow the instructions in make.doc. You will need 5 or 6 megs of space to build it in.) Then type '`gs -dNOPAUSE -sDEVICE=`*??????* `-sOutputFile=/dev/`*????* `file.ps`' and your output should (hopefully) appear at your printer. Those of you with non-US paper sizes may wish to build gs yourself with the right default, or you may use the '`-sPAPERSIZE=a4`' option.

Ghostscript may be used to print at most of the resolutions your printer supports; '`-r300`', '`-r150`', '`-r360x180`' are examples of the option used to control this. Dot matrix printers in particular need to choose an appropriate resolution, since they do not typically run at the more standard 300dpi. Note that versions 2.6 and greater of ghostscript have more drivers.

2.4 fonts

(This section contains font information not specific to any ghostscript driver, nor even specific to ghostscript). Font information pertaining to the X11 `gs` drivers (and thus to ghostview) is included in the ghostview section under previewing)

All versions of ghostscript come with assorted public-domain fonts, most of which were generated from bitmaps, and are therefore of relatively poor quality. However, ghostscript can use any PostScript Type 1 or 3 fonts you may have handy. For example, the Adobe Type Manager (for any platform other than mac) comes with fonts you can use. Place the fonts (typically *.pc?) in `lib/ghostscript/fonts/` and add to `lib/ghostscript/Fontmap` lines such as:

```
/Courier  (com_____.pfb) ;
```

```
# *     sparc   SPARCprinter
# *     t4693d2  Tektronix 4693d color printer, 2 bits per R/G/B component
# *     t4693d4  Tektronix 4693d color printer, 4 bits per R/G/B component
# *     t4693d8  Tektronix 4693d color printer, 8 bits per R/G/B component
# *     tek4696  Tektronix 4695/4696 inkjet plotter
#%*     trufax  TruFax facsimile driver  [Unix only]
# File formats and others:
#       bit     A plain "bit bucket" device
#       bmpmono Monochrome MS Windows .BMP file format
#       bmp16   4-bit (EGA/VGA) .BMP file format
#       bmp256  8-bit (256-color) .BMP file format
#       bmp16m  24-bit .BMP file format
#       gifmono Monochrome GIF file format
#       gif8    8-bit color GIF file format
#       pcxmono Monochrome PCX file format
#       pcxgray 8-bit gray scale PCX file format
#       pcx16   Older color PCX file format (EGA/VGA, 16-color)
#       pcx256  Newer color PCX file format (256-color)
#       pbm     Portable Bitmap (plain format)
#       pbmraw  Portable Bitmap (raw format)
#       pgm     Portable Graymap (plain format)
#       pgmraw  Portable Graymap (raw format)
#       ppm     Portable Pixmap (plain format)
#       ppmraw  Portable Pixmap (raw format)
# *     tiffg3  TIFF/F (G3 fax)
```

Lines beginning with a '#' are drivers included with the gs261 source distribution.

Drivers marked with '%' require commercial software to work

Installations of ghostscript are available from several places:

A full, and presumably correct, installation comes with most distributions. Note that this binary installation may require `libX.so.???`, as it includes the X11 display driver.

The "official" installation is to obtain the sources and build it yourself:

```
   prep.ai.mit.edu:/pub/gnu/ghostscript-xxxx.tar.gz
prep.ai.mit.edu:/pub/gnu/ghostscript-fonts-xxxx.tar.gz
```

This is probably best, as you can get the latest version (currently 2.6.1pl4 (the four patches are separate, they are available from my mail-server as `gs261-patches`)).

PostScript input and generates output appropriate for X displays, most printers, some specialized display hardware, and fax software.

The following devices are available as of ghostscript 2.6.1pl4:

```
        linux   PC vga using linux svgalib
#       x11     X Windows version 11, release >=4   [Unix and VMS only]
# Printers:
# *     appledmp  Apple Dot Matrix Printer (Imagewriter)
#       bj10e   Canon BubbleJet BJ10e
# *     bj200   Canon BubbleJet BJ200
# *     cdeskjet  H-P DeskJet 500C with 1 bit/pixel color
# *     cdjcolor  H-P DeskJet 500C with 24 bit/pixel color and
#               high-quality color (Floyd-Steinberg) dithering
# *     cdjmono  H-P DeskJet 500C printing black only
# *     cdj500  H-P DeskJet 500C (same as cdjcolor)
# *     cdj550  H-P DeskJet 550C
# *     declj250  alternate DEC LJ250 driver
# +     deskjet  H-P DeskJet and DeskJet Plus
# *     dfaxhigh  DigiBoard, Inc.'s DigiFAX software format
# *     dfaxlow  DigiFAX low (normal) resolution
#       djet500  H-P DeskJet 500
# *     djet500c  H-P DeskJet 500C
#       epson   Epson-compatible dot matrix printers (9- or 24-pin)
# +     eps9high  Epson-compatible 9-pin, interleaved lines
#               (triple resolution)
# *     epsonc  Epson LQ-2550 and Fujitsu 3400/2400/1200 color printers
# *     escp2   Epson ESC/P 2 language printers, including Stylus 800
# +     ibmpro  IBM 9-pin Proprinter
# *     jetp3852  IBM Jetprinter ink-jet color printer (Model #3852)
# +     laserjet  H-P LaserJet
# *     la50    DEC LA50 printer
# *     la75    DEC LA75 printer
# *     lbp8    Canon LBP-8II laser printer
# *     ln03    DEC LN03 printer
# *     lj250   DEC LJ250 Companion color printer
# +     ljet2p  H-P LaserJet IId/IIp/III* with TIFF compression
# +     ljet3   H-P LaserJet III* with Delta Row compression
# +     ljet4   H-P LaserJet 4 (defaults to 600 dpi)
# +     ljetplus  H-P LaserJet Plus
# *     m8510   C.Itoh M8510 printer
# *     necp6   NEC P6/P6+/P60 printers at 360 x 360 DPI resolution
# *     nwp533  Sony Microsystems NWP533 laser printer   [Sony only]
# *     oki182  Okidata MicroLine 182
#       paintjet  H-P PaintJet color printer
# *     pj      alternate PaintJet XL driver
# *     pjxl    H-P PaintJet XL color printer
# *     pjxl300  H-P PaintJet XL300 color printer
# *     r4081   Ricoh 4081 laser printer
```

2 Printing software

These sections describe printing software known to be available for Linux. Note that most Un*x printing software can be easily compiled under Linux.

2.1 text

Under Linux, or under most any Un*x operating system, the easiest thing to print with most printers is plain ASCII. Any way you can arrange to send the text to the printer is perfectly valid. If you have a serial printer, then try devices `/dev/ttyS?`, `/dev/ttys?`, `/dev/cua?`, etc, and if you have a regular parallel hookup, use `/dev/lp?`. Typing '`cat` *file* `>/dev/????`' should do it. You may need root privileges, or you may wish to chmod your printer device if this is all you ever print. (Note the security hole there if you've more than one user) Some printers (ie, HP DeskJet) want dos-style end of lines: newline+carriage return. These printers will show the 'staircase effect'. See Section 3.11 [The staircase effect], page 34.

2.2 pr

Most plain ascii files in the un*x world have a tendency to be just that - plain, unformatted ascii with no page breaks or anything else to make a printed copy look nice and not have lines printed over perferations. The answer to this problem is to run your text through a formatter such as pr. Pr is a standard un*x utility designed to format plain text for printing on a line printer. The usual appearance of the resulting formatted text has a header and/or footer, page numbers, the date, possibly margins, double spacing, etc. As is common with un*x utils, pr has a bazillion options. They are detailed in the man page.

2.3 PostScript

Printing almost anything other than plain text under Un*x usually involves the ability to print PostScript. If you have a PostScript printer, you're all set. But for most, this is not so easy.

The established way to interpret PostScript under Linux is to use ghostscript, which, like nearly everything else, comes from the GNU project. Ghostscript is a PostScript interpreter which accepts

Attributions on individual sections do not appear in the final version although they do in the texinfo source.

Assorted little changes

Incorporated Karl Auer's <Karl.Auer@anu.edu.au> document. This is still not seemless so you'll likely find some duplication and missing cross references - please let us know.

1 Intro

This is the Printing HOWTO for Linux. This is a member of the second generation of Linux FAQs. The original Linux FAQ gradually became a monolithic beast, and has now been re-written in a new syle, refering to HOWTOs for details. This HOWTO details how to properly set up most types of printers, how to configure software to make them print well, what types of fax software are available, and how to preview many types of printable output. It was originally written by Grant Taylor <gtaylor@cs.tufts.edu> and later incorporated the lpd-FAQ by Brian McCauley <B.A.McCauley@bham.ac.uk>. Other HOWTOs detail networking, kernel hacking, etc. All HOWTOs are found in several places, the official location being:
`sunsite.unc.edu:/pub/Linux/docs/HOWTO`

Like just about any usenet FAQs they can also be found on `rtfm.mit.edu`.

Please send comments, bug reports, etc, to <printing@god.ext.tufts.edu>. This address goes to both Brian and Grant.

If you know anything or can point us towards any useful programs not in this document that should be, please let us know!

Grant has a mail server set up on his machine. It stores documents and other things related to printing and previewing with Linux, including the latest and greatest version of this document. At some point in the future this will probably be superceded by our own directory on one of the major ftp servers. Mail to <listserv@god.ext.tufts.edu> with a body of '`info`' will have you sent a list of available files. A body of '`get file1 [file2 file3... ]`' will send you specific files.

Summary of changes

Now in texinfo! Available from sunsite in texinfo, dvi and text forms. The info form may be read in emacs but less you are a real emacsphile or don't have X then you'd probably be better off using `tkinfo`.

`tkinfo` is available by anonymous ftp from: `ptolemy.eecs.berkeley.edu:pub/misc`
`harbor.ecn.purdue.edu:pub/tcl/code`

The ptolemy site will always have the latest version.

New mailserver info.

Much dead-wood removed.

Linux Printing HOWTO guide

Grant Taylor <gtaylor@cs.tufts.edu>
Brian McCauley <B.A.McCauley@bham.ac.uk>