The Linux Printing HOWTO

Table of Contents

Table of Contents

Table of Contents

The Linux Printing HOWTO

Grant Taylor <gtaylor+pht@picante.com>

Version 4.5, Feb 2000

This is the Linux Printing HOWTO, a collection of information on how to generate, preview, print and fax anything under Linux (and other Unices in general).

5.Which spooling software?

6.How it works, basic

7.How to set things up

8.Vendor Solutions

9.Ghostscript.

10.How to print to a printer over the network

1.Introduction

The Printing HOWTO should contain everything you need to know to help you set up printing services on your Linux box(en). As life would have it, it's a bit more complicated than in the point−and−click world of Microsoft and Apple, but it's also a bit more flexible and certainly easier to administer for large LANs.

This document is structured so that most people will only need to read the first half or so. Most of the more

obscure and situation−dependent information in here is in the last half, and can be easily located in the Table of Contents, whereas most of the information through section 8 or 9 is probably needed by most people.

Since version 3.x is a complete rewrite, much information from previous editions has been lost. This is by design, as the previous HOWTOs were so large as to be 60 typeset pages, and had the narrative flow of a dead turtle. If you do not find the answer here, you are encouraged to a) scan the previous version at the Printing HOWTO Home Page and b) drop me a note saying what ought to be here but isn't.

The Printing HOWTO Home Page is a good place to find the latest version; it is also, of course, distributed from Metalab (`metalab.unc.edu`) and your friendly local LDP mirror.

1.1 History

This is the fourth generation of the Printing HOWTO. The history of the PHT may be chronicled thusly:

1. I wrote the printing−howto in 1992 in response to too many printing questions in comp.os.linux, and posted it. This predated the HOWTO project by a few months and was the first FAQlet called a `howto'. This edition was in plain ascii.
2. After joining the HOWTO project, the Printing−HOWTO was merged with an Lpd FAQ by Brian McCauley <B.A.McCauley@bham.ac.uk>; we continued to co−author the PHT for two years or so. At some point we incorporated the work of Karl Auer <Karl.Auer@anu.edu.au>. This generation of the PHT was in TeXinfo, and available in PS, HTML, Ascii, and Info.
3. After letting the PHT rot and decay for over a year, and an unsuccessful attempt at getting someone else to maintain it, this rewrite happened. This generation of the PHT is written in SGML using the LinuxDoc DTD and the SGML−Tools−1 package. Beginning with version 3.27, it incorporates a summary of a companion printer support database; before 3.27 there was never a printer compatability list in this HOWTO (!).
4. In mid−January, 2000, I found out about the PDQ print "spooler". PDQ provides a printing mechanism so much better than lpd ever did that I spent several hours playing with it, rewrote parts of this HOWTO, and bumped the version number of the document to 4.

1.2 Copyright

2.How to print

You actually use a different command to print depending on which spooling software you use.

2.1 With PDQ

Most systems today ship with lpd, so this section won't apply. That said, I now recommend that people install and use PDQ in most cases instead of (or in addition to) lpd. PDQ just has much better support for printer options and such.

With PDQ, instead of the lpr command, you use the command pdq or xpdq. Both work much like the traditional lpr in that they will print the files you specify, or stdin if no files are given.

Xpdq

Xpdq is an X Windows application that shows a list of available printers and a summary of the print queue (including current and historical jobs). There are two options under the File menu, one to print specific files, and one to print stdin. You can set whatever options are defined in your printer driver from the Driver Options dialog; typically there will be duplex, resolution, paper type and size settings, and so forth.

Pdq

The PDQ system's command−line printing command is simply called pdq. It can be used in place of the lpr command in most situations; it accepts the `−P` printer specification argument. Like lpr, it prints either the listed file(s) or stdin.

Printer options can be controlled with the `−o` and `−a` options.

2.2 With LPD and the lpr command

If you've already got lpd setup to print to your printer, or your system administrator already did so, or your vendor did so for you, then all you need to do is learn how to use the lpr command. The Printing Usage HOWTO covers this, and a few other queue manipulation commands you should probably know. Or just read the lpr(1) man page.

If, however, you have a new system or new printer, then you'll have to set up printing services one way or another before you can print. Read on!

3.Kernel printer devices

There are two completely different device drivers for the parallel port; which one you are using depends on your kernel version (which you can find out with the command `uname −a`). The driver changed in Linux 2.1.33.

A few details are the same for both styles of driver. Most notably, many people have found that Linux will

not detect their parallel port unless they disable "Plug and Play" in their PC BIOS. (This is no surprise; the track record for PnP of non−PCI devices with Windows and elsewhere has been something of a disaster).

3.1 The lp device (kernels <=2.1.32)

The Linux kernel (<=2.1.32), assuming you have compiled in or loaded the lp device (the output of `cat /proc/devices` should include the device lp if it is loaded), provides one or more of */dev/lp0*, */dev/lp1*, and */dev/lp2*. These are NOT assigned dynamically, rather, each corresponds to a specific hardware I/O address. This means that your first printer may be *lp0* or *lp1* depending on your hardware. Just try both.

A few users have reported that their bidirectional lp ports aren't detected if they use an older unidirectional printer cable. Check that you've got a decent cable.

One cannot run the plip and lp drivers at the same time on any given port (under 2.0, anyway). You can, however, have one or the other driver loaded at any given time either manually, or by kerneld with version 2.x (and later 1.3.x) kernels. By carefully setting the interrupts and such, you can supposedly run plip on one port and lp on the other. One person did so by editing the drivers; I eagerly await a success report of someone doing so with only a clever command line.

There is a little utility called `tunelp` floating about with which you, as root, can tune the Linux 2.0 lp device's interrupt usage, polling rate, and other options.

When the lp driver is built into the kernel, the kernel will accept an `lp=` option to set interrupts and io addresses:

```
When the lp driver is built in to the kernel, you may use the
LILO/LOADLIN command line to set the port addresses and interrupts
that the driver will use.

Syntax:      lp=port0[,irq0[,port1[,irq1[,port2[,irq2]]]]]

For example:   lp=0x378,0   or   lp=0x278,5,0x378,7 **

Note that if this feature is used, you must specify *all* the ports
you want considered, there are no defaults.  You can disable a
built−in driver with lp=0.
```

When loaded as a module, it is possible to specify io addresses and interrupt lines on the insmod command line (or in */etc/conf.modules* so as to affect kerneld) using the usual module argument syntax. The parameters are `io=port0,port1,port2` and `irq=irq0,irq1,irq2`. Read ye the man page for *insmod* for more information on this.

**For those of you who (like me) can never find the standard port numbers when you need them, they are as in the second example above. The other port (*lp0*) is at 0x3bc. I've no idea what interrupt it usually uses.

The source code for the Linux 2.0 parallel port driver is in /usr/src/linux/drivers/char/lp.c.

3.2 The parport device (kernels >= 2.1.33)

Beginning with kernel 2.1.33 (and available as a patch for kernel 2.0.30), the lp device is merely a client of the new parport device. The addition of the parport device corrects a number of the problems that plague the old lp device driver − it can share the port with other drivers, it dynamically assigns available parallel ports to device numbers rather than enforcing a fixed correspondence between I/O addresses and port numbers, and so forth.

The advent of the parport device has enabled a whole flock of new parallel−port drivers for things like Zip drives, Backpack CD−ROMs and disks, and so forth. Some of these are also available in versions for 2.0 kernels; look around on the web.

The main difference that you will notice, so far as printing goes, is that parport−based kernels dynamically assign lp devices to parallel ports. So what was lp1 under Linux 2.0 may well be lp0 under Linux 2.2. Be sure to check this if you upgrade from an lp−driver kernel to a parport−driver kernel.

The most popular problems with this device seems to stem from misconfiguration:

The Distribution

Some Linux distributions don't ship with a properly setup /etc/modules.conf (or /etc/conf.modules), so the driver isn't loaded properly when you need it to be. With a recent modutils, the proper magical lines from modules.conf seem to be:

```
alias /dev/printers lp            # only for devfs?   alias /dev/lp*      lp
```

The BIOS

Many PC BIOSes will make the parallel port into a Plug−and−Play device. This just adds needless complexity to a perfectly simple device that is nearly always present; turn off the PnP setting for your parallel prot ("LPT1" in many BIOSes) if your parallel port isn't detected by the Linux driver. The correct setting is often called "legacy", "ISA", or "0x378", but probably not "disabled".

You can also read the file Documentation/parport.txt in your kernel sources, or look at the parport web site.

3.3 Serial devices

Serial devices are usually called something like */dev/ttyS1* under Linux. The utility `stty` will allow you to interactively view or set the settings for a serial port; `setserial` will allow you to control a few extended attributes and configure IRQs and I/O addresses for non−standard ports. Further discussion of serial ports under Linux may be found in the Serial−HOWTO.

When using a slow serial printer with flow control, you may find that some of your print jobs get truncated. This may be due to the serial port, whose default behavior is to purge any untransmitted characters from its buffer 30 seconds after the port device is closed. The buffer can hold up to 4096 characters, and if your printer uses flow control and is slow enough that it can't accept all the data from the buffer within 30 seconds

after printing software has closed the serial port, the tail end of the buffer's contents will be lost. If the command `cat file > /dev/ttyS2` produces complete printouts for short files but truncated ones for longer files, you may have this condition.

The 30 second interval can be adjusted through the "closing_wait" commandline option of setserial (version 2.12 and later). A machine's serial ports are usually initialized by a call to setserial in the rc.serial boot file. The call for the printing serial port can be modified to set the closing_wait at the same time as it sets that port's other parameters.

4.Supported Printers

The Linux kernel mostly supports any printer that you can plug into a serial or parallel port, but there are things to look out for, and printers that you won't be able to use, even though they can (electrically speaking) communicate with Linux. Primary among these incompatible printers are those referred to as "Windows" or "GDI" printers. They are called this because part or all of the printer control language and the design details of the printing mechanism are not documented. Typically the vendor will provide a Windows driver and happily sell only to Windows users; this is why they are called Winprinters. In some cases the vendor also provides drivers for NT, OS/2, or other operating systems.

Many of these printers *do not work* with Linux. A few of them do, and some of them only work a little bit (usually because someone has reverse engineered the details needed to write a driver). See the printer support list below for details on specific printers.

A few printers are in−between. Some of NEC's models, for example, implement a simple form of the standard printer language PCL that allows PCL−speaking software to print at up to 300dpi, but only NEC knows how to get the full 600dpi out of these printers.

Note that if you already have one of these Winprinters, there are roundabout ways to get Linux to print to one, but they're rather awkward and I've never tried it myself. See Section 12 of this document for more discussion of Windows−only printers.

4.1 Postscript

As for what printers *do* work with Linux, the best choice is to buy a printer with native PostScript support. Nearly all Unix software that produces printable output produces it in PostScript, so obviously it'd be nice to get a printer that supports PostScript directly. Unfortunately, PostScript support is scarce outside the laser printer domain, and is sometimes a costly add−on.

Unix software, and the publishing industry in general, have standardized upon Postscript as the printer control language of choice. This happened for several reasons:

Timing

Postscript arrived as part of the Apple Laserwriter, a perfect companion to the Macintosh, the system largely responsible for the desktop publishing revolution of the 80s.

It's device−independent

Postscript programs can be run to generate output on a pixel screen, a vector screen, a fax machine, or almost any sort of printer mechanism, without the original program needing to be changed. Postscript output will look the same on any Postscript device, at least within the limits of the device's capabilities. Before the creation of PDF, people exchanged complex documents online as Postscript files. The only reason this standard didn't "stick" was because Windows machines didn't usually include a Postscript previewer, so Adobe specified hyperlinks and compression for Postscript, called the result PDF, distributed previewers for it, and invented a market for their "distiller" tools (the functionality of which is also provided by ghostscript's ps2pdf and pdf2ps programs).

It's a real programming language

Postscript is a complete programming language; you can write software to do most anything in it. This is mostly useful for defining subroutines at the start of your program to reproduce complex things over and over throughout your document, like a logo or a big "DRAFT" in the background.

It's open

Postscript is fully specified in a publically available series of books (which you can find at any good bookstore). Although Adobe invented it and provides the dominant commercial implementation, other vendors like Aladdin produce independently coded implementations as well.

4.2 Non−Postscript

Failing the (larger) budget necessary to buy a Postscript printer, you can use any printer supported by Ghostscript, the free Postscript interpreter used in lieu of actual printer Postscript support. Note that most Linux distributions can only ship a somewhat outdated version of Ghostscript due to the license. Fortunately, there is usually a prepackaged up to date Ghostscript made available in each distribution's contrib area. Please help improve the Ghostscript printer support page by reporting your successes and failures as it asks.

Adobe now has a new printer language called "PrintGear". I think it's a greatly simplified binary format language with some Postscript heritage but no Postscript compatibility. And I haven't heard of Ghostscript supporting it. But some PrintGear printers seem to support another language like PCL, and these printers will work with Linux (iff the PCL is implemented in the printer and not in a Windows driver).

4.3 What printers work?

If you want to buy a printer, you can look in several places to see if it will work. The cooperatively maintained Printing HOWTO printer database aims to be a comprehensive listing of the state of Linux printer support. A summary of it is below; be sure to check online for more details and information on what driver to use.

Ghostscript's printer compatibility page has a list of some working printers, as well as links to other pages.

Dejanews contains hundreds of "it works" and "it doesn't work" testimonials. Try all three, and when you're done, check that your printer is present and correct in the database, so that it will be listed properly in this document in the future.

Printer compatibility list

This section is a summary of the online version. The online version includes basic specifications, notes, links to driver information, user−maintained documentation, manufacturer web pages, and so forth. The online version of this list is also interactive; people can and do add printers all the time, so be sure to check it as well. Finally, if your printer isn't listed, add it!

Note that this listing is not gospel; people sometimes add incorrect information, which I eventually weed out. Entries I have not sanity−checked are merked with an asterisk (*). Verify from Dejanews that a printer works for someone before buying it based on this list. If you can find no information in Dejanews, mail me and I'll put you in contact with the person who added the printer.

Printers here are categorized into three types:

Perfectly

Perfect printers work perfectly − you can print to the full ability of the printer, including color, full resolution, etc. In a few cases printers with undocumented "resolution enhancement" modes that don't work are listed as perfect; generally the difference in print quality is small enough that it isn't worth worrying about.

Mostly

You can print fine, but there may be minor limitations or one sort or another in either printing or other features.

Partially

You can print, but maybe not in color, or only at a poor resolution. See the online listing's notes column for information on the limitation.

Paperweight

You can't print a darned thing; typically this will be due to lack of a driver and/or documentation on how to write one.

In all cases, since this information is provided by dozens of people, none of it is guaranteed to be correct; entries with an asterisk (*) are particularly suspect. The facts, however, should be easy to corroborate from the driver web pages and manufacturer web sites.

And without further ado, here is the printer compatibility list:

Alps

Partially

MD−1000, MD−1300, MD−2000, MD−4000, MD−5000.

Apple

Perfectly

Dot Matrix, ImageWriter*, ImageWriter LQ, LaserWriter 16/600, LaserWriter IINTX*, LaserWriter Select 360.

Mostly

12/640ps, LaserWriter NT, StyleWriter 2500.

Avery

Perfectly

Personal Label Printer+.

Mostly

Personal Label Printer.

Brother

Perfectly

HL−1070, HL−10V, HL−10h, HL−1260, HL−2060, HL−4Ve, HL−630*, HL−720*, HL−720*, HL−730, HL−760, HL−8*, HL−820.

Mostly

HJ−400, HL−1040, HL−1050, HL−1060, HL−1240*, HL−1250, MFC 6550MC, MFC4350*.

Partially

MC−3000, MFC 7150C, MFC8300*.

Paperweight

HL−1030, MP−21C.

C.Itoh

Perfectly

M8510.

CalComp

Paperweight

Artisan 1023 penplotter*.

Canon

Perfectly

BJ−10e, BJ−20, BJ−200, BJ−330, BJ−5, BJC−210, BJC−250, BJC−4000, BJC−4100, BJC−4200, BJC−4300, BJC−4400, BJC−600, BJC−610, BJC−620*, BJC−70, BJC−800, GP335/405*, LBP−1260*, LBP−1760, LBP−4+*, LBP−4U*, LBP−8A1*, LIPS III*, LIPS−III*, bjc5000*.

Mostly

BJ−300*, BJC−1000, BJC−2000, BJC−210SP*, BJC−240, BJC−4310SP*, BJC−7004*, BJC−80, LBP−4sx*.

Partially

BJC−4550*, BJC−6000, BJC−7000*, BJC−7100*, MultiPASS C2500*, MultiPASS C3500*, MultiPASS C5000*, Multipass C3000*, Multipass C5500*.

Paperweight

BJC−5000, BJC−5100, LBP−430, LBP−460*, LBP−660*, Multipass L6000*.

Citizen

Perfectly

ProJet II*, ProJet IIc*.

Partially

printiva600C*.

DEC

Perfectly

DECWriter 500i*, DECwriter 110i*, DECwriter 520ic*, LA50*, LA75*, LA75 Plus*, LN03*, LN07*.

Mostly

LJ250*, LN17.

Partially

1800*.

Dymo−CoStar

Perfectly

ASCII 250*, ASCII+*, EL40*, EL60*, LabelWriter II*, LabelWriter XL*, LabelWriter XL+*, SE250*, SE250+*, Turbo*.

Epson

Perfectly

9 Pin Printers high−res*, 9 Pin Printers med−res*, AP3250*, ActionLaser 1100*, LP 8000*, LQ 850*, LQ−24*, LQ−2550*, LQ−500*, LQ−570+*, LX−1050*, SQ 1170*, Stylus Color*, Stylus Color 1520, Stylus Color 400*, Stylus Color 440, Stylus Color 460*, Stylus Color 500*, Stylus Color 600*, Stylus Color 640*, Stylus Color 800*, Stylus Color 850*, Stylus Color I*, Stylus Color II*, Stylus Color IIs*, Stylus Color PRO*, Stylus Pro XL*.

Mostly

EPL 5700*, Stylus 300*, Stylus Color 3000*, Stylus Color 660, Stylus Color 740*.

Partially

Stylus Color 300*, Stylus Color 900*, Stylus Photo 700*, Stylus Photo 750*, Stylus Photo EX*.

Fujitsu

Perfectly

1200*, 2400*, 3400*, PrintPartner 10V*, PrintPartner 16DV*, PrintPartner 20W*, PrintPartner 8000*.

HP

Perfectly

2000C*, 2500C, Color LaserJet 4500, DeskJet 1200C, DeskJet 1200C/PS, DeskJet 1600C, DeskJet 1600Cm, DeskJet 400, DeskJet 420C, DeskJet 500, DeskJet 500C*, DeskJet 510*, DeskJet 520*, DeskJet 540*, DeskJet 550C*, DeskJet 560C*, DeskJet 600*, DeskJet 610C*, DeskJet 610CL*, DeskJet 612C*, DeskJet 660C*, DeskJet 670C*, DeskJet 672C*, DeskJet 682C*, DeskJet 690C*, DeskJet 692C*, DeskJet 694C*, DeskJet 697C*, DeskJet 812C*, DeskJet 850C, DeskJet 855C*, DeskJet 890C, HP LaserJet 2P Plus*, LaserJet*, LaserJet 1100*, LaserJet 1100A*, LaserJet 2 w/PS*, LaserJet 2100M*, LaserJet 2D*, LaserJet 2P*, LaserJet 3*, LaserJet 3D*, LaserJet 3P w/PS*, LaserJet 4 Plus*, LaserJet 4050N*, LaserJet 4L*, LaserJet 4M*, LaserJet 4ML*, LaserJet 4P*, LaserJet 5*, LaserJet 5000*, LaserJet 5L*, LaserJet 5M*, LaserJet 5MP*, LaserJet 5P*, LaserJet 6*, LaserJet 6MP*, LaserJet 8000*, LaserJet 8100*, LaserJet Plus*, Mopier 320*, PaintJet*, PaintJet XL*, PaintJet XL300*.

Mostly

DesignJet 650C*, Designjet 750 C Plus*, DeskJet 1100C*, DeskJet 1120C*, DeskJet 310, DeskJet 810C, DeskJet 832C*, DeskJet 870C*, DeskJet 880C*, DeskJet 882C, DeskJet 895C*, DeskJet 895Cxi*, DeskJet 970C*, DeskJet 970Cse, LaserJet 2*, LaserJet 2100*, LaserJet 6P*, OfficeJet Pro 1170Cse*.

Partially

Color LaserJet 5000, DeskJet 1000C*, DeskJet 710C*, DeskJet 712C*, DeskJet 720C*, DeskJet 722C*, DeskJet 820C*, LaserJet 6L*, OfficeJet 500*, OfficeJet 600*, OfficeJet 625*, OfficeJet Pro 1175C*, PhotoSmart P1100*.

Paperweight

LaserJet 3100*.

IBM

Perfectly

3853 JetPrinter*, 4019*, 4029 10P*, 4303 Network Color Printer*, Page Printer 3112*, ProPrinterII*.

Imagen

Perfectly

ImPress*.

Kyocera

Perfectly

F−3300*, FS−1700+*, FS−3750*, FS−600*, FS−800*, P−2000*.

Mostly

FS−3500*.

Lexmark

Perfectly

4039 10plus*, Optra Color 1200*, Optra Color 1275*, Optra Color 40, Optra Color 45, Optra E*, Optra E+*, Optra E310*, Optra Ep*, Optra K 1220*, Optra R+*, Optra S 1250*, Optra S 1855*, Valuewriter 300*.

Mostly

1000, 1100*, 2070*, 3000*, 5000*, 5700, 7000*, 7200*.

Partially

1020 Business*, 2030*, Winwriter 400*, Z51*.

Paperweight

1020*, 2050*, 3200*, Winwriter 100*, Winwriter 150c*, Winwriter 200*, Z11*.

Minolta

Perfectly

PagePro 6*, PagePro 6e*, PagePro 6ex*, PagePro 8*.

Partially

PagePro 8L*.

Mitsubishi

Perfectly

CP50 Color Printer*.

NEC

Perfectly

P2X*, PinWriter P6*, PinWriter P6 plus*, PinWriter P60*, PinWriter P7*, PinWriter P7 plus*, PinWriter P70*, SilentWriter LC 890*, Silentwriter2 S60P*, Silentwriter2 model 290*, SuperScript 660i*.

Mostly

Silentwriter 95f*.

Partially

SuperScript 100C*, SuperScript 1260*, SuperScript 150C*, SuperScript 650C*, SuperScript 750C*, SuperScript 860*, SuperScript 870.

Paperweight

SuperScript 610plus*, SuperScript 660*, SuperScript 660plus*.

Oce

Perfectly

3165*.

Okidata

Perfectly

OL 410e, OL 600e, OL 610e/PS, OL 800, OL 810e/PS, OL400ex, OL810ex, OL830Plus, Okipage 10e, Okipage 12i, Okipage 20DXn, Okipage 6e, Okipage 6ex, Okipage 8c, Okipage 8p.

Mostly

Microline 182, OL 400w, OL 610e/S, OkiPage 4w+*, Okipage 4w, Super 6e.

Partially

Microline 192+, Okipage 6w.

Paperweight

Okijet 2010, Okijet 2500, Okipage 8w*.

Olivetti

Perfectly

JP350S*, JP450*, PG 306*.

PCPI

Perfectly

1030*.

Panasonic

Perfectly

KX−P1123*, KX−P1124*, KX−P1150*, KX−P1180i*, KX−P2023*, KX−P2135*, KX−P2150*, KX−P4410, KX−P4450*, KX−P5400*, KX−P8420*, KX−P8475*, kx−p1624*.

Mostly

KX−P2123*, KX−P6150*.

Partially

KX−P6500*.

Paperweight

KX−P6100*, KX−P6300 GDI*, KX−P8410*.

Printrex

Partially

820 DL*.

QMS

Perfectly

2425 Turbo EX*, magicolour 2*.

Mostly

ps−810*.

Ricoh

Perfectly

4081*, 4801*, 6000*, Aficio AP2000*.

Mostly

Aficio 401*.

Paperweight

Aficio Color 2206*, Afico FX10*.

Samsung

Perfectly

ML−5000a*, ML−6000/6100*, ML−7000/7000P/7000N*, ML−7050*, ML−85*, QL−5100A*.

Mostly

ML−5050G*.

Paperweight

ML−85G*, SF/MSYS/MJ−4700/4800/4500C*.

Seiko

Perfectly

SpeedJET 200*.

Mostly

SLP*, SLP 120*, SLP 220*, SLP EZ30*, SLP Plus*, SLP Pro*.

Sharp

Perfectly

AR−161*.

Star

Perfectly

LC24−100*, NL−10*.

Mostly

LC 90*, LC24−200*, StarJet 48*.

Paperweight

WinType 4000*.

Tally

Perfectly

MT908*.

Tektronix

Perfectly

3693d color printer, 8−bit mode*, 4693d color printer, 2−bit mode*, 4693d color printer, 4−bit mode*, 4695*, 4696*, 4697*, Phaser 780*, Phaser IISX*, Phaser PX*.

Xerox

Perfectly

2700 XES, 3700 XES, 4045 XES, DocuPrint 4508, DocuPrint C55, DocuPrint N17, DocuPrint N32.

Mostly

DocuPrint P12, DocuPrint P8e, XJ6C*.

Partially

Document Homecentre, WorkCentre 450cp*, XJ8C*.

Paperweight

DocuPrint P8, WorkCentre 470cx*, WorkCentre XD120f*, WorkCentre XE80. * This entry has not been sanity−checked by me.

4.4 How to buy a printer

It's a bit dificult to select a printer these days; there are many models to choose from. Here are some shopping tips:

Cost

You get what you pay for. Most printers under $200−300 will print reasonably well, but printing costs a lot per page. For some printers, it only takes one or two cartridges to add up to the cost of a

new printer! Similarly, the cheapest printers won't last very long. The least expensive printers, for example, have a MTBF of about three months.

Inkjets

Inkjet printheads will clog irreparably over time, so the ability to replace the head somehow is a feature. Inkjet printheads are expensive, with integrated head/ink cartridges costing ten times (!) what ink−only cartridges go for, so the ability to replace the head only when needed is a feature. Epson Styluses tend to have fixed heads, and HP DeskJets tend to have heads integrated into the cartridges. Canons have three−part cartridges with independently replaceable ink tanks; I like this design. OTOH, the HP cartridges aren't enormously more expensive, and HP makes a better overall line; Canon is often the third choice from the print quality standpoint. You can't win.

Lasers

Laser printers consume a drum and toner. The cheapest designs include toner and drum together in a big cartridge; these designs cost the most to run. The best designs for large volume take plain toner powder or at least separate toner cartridges and drums.

Photography

The best color photograph output is from continuous tone printers like the Alps series (thermal transfer of dry ink or dye sublimation). A few of the Alps units are actually affordable, but they have poor Linux support (the one report I have speaks of banding and grainy pictures). The more common photo−specialized inkjets usually feature 6 color CMYKcm printing or even a 7 color CMYKcmy process; only models with Postscript support work with Linux, since Ghostscript doesn't seem to support 6 and 7 color printing. Good CMYK output is nothing to sneeze at, though. All photo−specialized printers are expensive to run; either you always run out of blue and have to replace the whole cartridge, or the individual color refills for your high−end photo printer cost an arm and a leg. Special papers cost a bundle, too. See also the section on printing photographs later in this document, and the sections on color tuning in Ghostscript.

Speed

Speed is proportional to processing power, bandwidth, and generally printer cost. The fastest printers will be networked postscript printers with powerful internal processors. Consumer−grade printers will depend partly on Ghostscript's rendering speed, which you can affect by having a reasonably well−powered machine; full pages of color, in particular, can consume large amounts of host memory.

Forms

If you want to print on multicopy forms, then you need an impact printer; many companies still make dot matrix printers, most of which emulate traditional Epson models and thus work fine.

Labels

There are two supported lines of label printer; look for the Dymo−Costar and the Seiko SLP models. Other models may or may not work. Avery also makes various sizes of stick−on labels in 8.5x11 format that you can run through a regular printer.

Plotting

Big drafting formats are usually supported these days by monster inkjets; HP is a popular choice. Mid−sized (11x17) inkjets are also commonly used for smaller prints. Much plotting of this sort is done with the languages RTL, HP−GL, and HP−GL/2, all of which are simple HP proprietary vector languages usually generated directly by application software.

What do I have?

I own an HP Deskjet 500 and a Lexmark Optra 40. Both work perfectly: the Deskjet is an older monochrome model, well−supported by Ghostscript; and the Optra is a more modern color inkjet with full Postscript and PCL 5 support (!).

I also own a Hawking Technology 10/100 Ethernet print server (model 7117, actually made by Zero One Technologies in Taiwan); this makes it possible to put the printer anywhere with power and a network jack, instead of just near a computer. It's a little dongle that attaches to the printer's parallel port and has an Ethernet jack on the other side. The only flaw with this is that it doesn't allow bidirectional communication, so I can't arrange to be sent email when the ink is low.

5.Which spooling software?

Until recently, the choice for Linux users was simple − everyone ran the same old lpd lifted mostly verbatim out of BSD's Net−2 code. Even today, most vendors ship this software. But this is beginning to change. SVR4−like systems including Sun's Solaris come with a completely different print spooling package, centered around lpsched.

Today, I recommend the PDQ system for both simple home users and (in a hybrid pdq/lpd setup) people in many larger environments. It provides both the simplest and most flexible configuration mechanism, and the nicest user utilities (indeed, it's the only one that provides a uniform printer option control vaguely equivalent to the functionality of the Print Setup dialogs in Windows).

5.1 PDQ

PDQ is a non−daemon−centric print system which has a built−in, and sensible, driver configuration syntax. This includes the ability to declare printing options, and a GUI or command line tool for users to specify these options with; users get a nice dialog box in which to specify resolution, duplexing, paper type, etc.

Running all of the filters as the user has a number of advantages: the security problems possible from Postscript are mostly gone, multi−file LaTeX jobs can be printed effectively as dvi files, and so forth. This is what I now use; I've written driver spec files for my printers, and there are several included with the distribution, so there are plenty of examples to base yours on. I've also written a few tools to automate driver spec generation to help the rest of you.

If you have many users, many printers, or anything else complex going on, I recommend using PDQ as a front−end to LPD−protocol based network printing (you can print via the lpd protocol to the local machine). In many such situations, rather than using BSD's lpd as the back−end, I recommend LPRng:

5.2 LPRng

There are signs that some Linux vendors will shift to providing LPRng, a far less ancient print spooling implementation that is more or less freely available. LPRng is far easier to administer for large installations (read: more than one printer, any serial printers, or any peculiar non−lpd network printers) and has a less frightfully haphazard codebase than does stock lpd. It can even honestly claim to be secure − there are no SUID binaries, and it supports authentication via PGP or Kerberos.

LPRng also includes some example setups for common network printers − HP LaserJets, mainly, that include some accounting abilities. If you'd like more information on LPRng, check out the LPRng Web Page.

LPRng is distributed under either the GPL or an Artistic license. (Previously that was not so, but it is now.)

5.3 PPR

PPR is a Postscript−centric spooler which includes a rudimentary Postscript parsing ability from which it derives several nice features. It includes good accounting capabilities, good support for Appletalk, SMB, and LPD clients, and much better error handling than lpd. PPR, like every other spooler here, can call Ghostscript to handle non−Postscript printers.

I only recently found out about PPR; I don't know of anyone who has tried it. It was written by, and is in use at, Trinity College. The license is BSD−style; free for all use but credit is due.

5.4 CUPS

One interesting newcomer on the scene is "CUPS", an implementation of the Internet Printing Protocol, an HTTP−esque RFC−defined replacement protocol for the venerable (and klunky) lpd protocol. The primary implementation of this is the open−source component of the commercial product "Easy Print", which consists of an intelligent spooler (CUPS) and a collection of commercial printer drivers built around Ghostscript (ESP Print Pro).

CUPS, the spooler, is distributed under a GPL license. ESP Print Pro is a binary−only commercial product (except for the included spooler, which is also available separately as the GPLed CUPS).

6.How it works, basic

In order to get printing working well, you need to understand how your spooling software works.

6.1 PDQ

Pdq stands for "Print, Don't Queue", and the way it works reflects this design. The following sequence of events happens when you use PDQ to print:

You run pdq or xpdq, specifying a file.

You specify a printer.

You specify the settings for the various options and arguments defined in the printer's PDQ driver file (duplex, copies, print quality, and so forth).

PDQ analyzes the contents of what you printed, and follows the instructions in the PDQ driver file which tell it how to process your data for this printer with your options.

PDQ sends the processed data to the printer according to the interface defined for that printer (straight to /dev/lp0, or to an LPD daemon on the network, over the network to an Apple or Microsoft system, or even to a fax machine).

If PDQ can't send the data to the printer right away, it spawns a background process to wait and try again until it succeeds or hits a time limit. At all times during this process, and afterwards, the state of each print job can be seen and inspected using xpdq. Jobs that failed are shown in red and can be resent.

6.2 LPD

Lpd stands for Line Printer Daemon, and refers in different contexts to both the daemon and the whole collection of programs which run print spooling. These are:

lpd

The spooling daemon. One of these runs to control everything on a machine, AND one is run per printer while the printer is printing.

lpr

The user spooling command. Lpr contacts lpd and injects a new print job into the spool.

lpq

Lists the jobs in a print queue.

lpc

The Lpd system control command. With lpc you can stop, start, reorder, etc, the print queues.

lprm

`lprm` will remove a job from the print spool.
So how does it fit together? Well, when the system boots, lpd is run. It scans the file */etc/printcap* to learn which printers it will be managing spools for. Each time someone runs lpr, lpr contacts lpd through the named socket */dev/printer*, and feeds lpd both the file to print and some information about who is printing and how to print it. Lpd then prints the file on the appropriate printer in turn.
The lp system was originally designed when most printers were line printers − that is, people mostly printed plain ascii. As it turns out, only a little extra scripting is needed to make lpd work quite well for today's print jobs, which are often in PostScript, or text, or dvi, or...

7.How to set things up

For common configurations, you can probably ignore this section entirely − instead, you should jump straight to the Vendor Solutions section below, or better yet, your vendor's documentation. Most Linux distributions supply one or more "idiot−proof" tools to do everything described here for common printers.
If your vendor's tool's results are not satisfactory, or you'd like the ability to interactively control printing options when you print, then you should use PDQ; I recommend PDQ in most cases.

7.1 Configuring PDQ

PDQ can be configured by either the superuser or by a joeuser. Root's changes are made to */etc/printrc*, and affect everyone, while joeuser can only modify his personal *.printrc*. Everything applies to both types of configuration.
If PDQ is not available prepackaged for your distribution, you should obtain the source distribution from the PDQ web page and compile it yourself. It is an easy compile, but you must first be sure to have installed the various GTK development library packages, the C library development package, the gcc compiler, make, and possibly a few other development things.

Drivers and Interfaces

PDQ lets users select a printer to print to. A printer is defined in PDQ as the combination of a "driver" and an "interface". Both drivers and interfaces are, in fact, merely snippets of text in the PDQ configuration file.
A PDQ interface says everything about how to ship data out to a printer. The most common interfaces, which are predefined in the PDQ distribution's example printrc file, are:

local−port

A local port interface speaks to a parallel or serial port on the machine PDQ is running on. Using this interface, PDQ can print directly to your parallel port. Note that if you have a multiuser system this can cause confusion, and if you have a network the local−port interface will only apply to one system. In those cases, you can define a raw unfiltered lpd queue for the port and print to the system's lpd daemon exactly the same way from all systems and accounts without any troubles. This interface has a device name argument; the typical value would be /dev/lp0.

bsd−lpd

A bsd lpd interface speaks over the network to an LPD daemon or LPD−speaking networked printer. PDQ supports job submission, cancellation, and queries to LPD interfaces. This interface has hostname and queuename arguments.

appletalk

The appletalk interface allows you to print to printers over the Appletalk network; if you have a printer plugged into your Mac this is the way to go. This interface needs to have the Netatalk package installed to work.

A PDQ driver says everything about how to massage print data into a format that a particular printer can handle. For Postscript printers, this will include conversion from ascii into Postscript; for non−Postscript printers this will include conversion from Postscript into the printer's language with Ghostscript.

If one of PDQ's included driver specifications doesn't fit your printer, then read the section below on how to write your own.

Defining Printers

To define a printer in PDQ:

First check that you've got suitable driver and interface declarations in the system or your personal printrc.

If you want to define the printer in /etc/printrc (for all users), then su to root.

Run xpdq, and select Printer−>Add printer. This "wizard" will walk you through the selection of a driver and interface. That's really all there is to it; most of the work lies in finding or creating a suitable driver specification if you can't find one premade.

Creating a PDQ Driver Declaration

Here I'll walk through an example of how to make a PDQ driver declaration. Before you try that, though, there are several places to look for existing driver specs:

PDQ itself comes with a small collection of prewritten driver files.

This HOWTO's database includes a program called "PDQ−O−Matic" which will generate a PDQ specification from the information in the database. With a bit of fiddling, this may suit. This is the easy path if you have a non−Postscript printer.

I've written a tool called `ppdtopdq` which takes a Postscript Printer Definition file and converts it into a PDQ driver specification. This is the obvious path if you have a Postscript printer. Mail me for a copy.

There are several places to look for the information needed to write your own PDQ driver:

The PDQ driver specification syntax is quite rich, and is fully documented in the printrc(5) man page.

The PDQ distribution includes a few example files. Look in particular at the Epson Stylus file, which demonstrates the structure of the definition for a Ghostscript−driven printer.

The Printing HOWTO Database includes raw Linux driver information for over 400 printers. This will tell you what options to give Ghostscript, or what extra program to run on the Ghostscript output.

If you have to create your own driver specification, or if you enhance one from the PDQ distribution or one of the PDQ driver generator programs mentioned above, please share your creation with the world! Send it to me (`gtaylor+pht@picante.com`), and I'll make sure that it gets found by future PDQ users with your type of printer.

Now, let's walk through the writing of a driver specification for a printer listed in the Printing HOWTO's database as working, but for which you can't find a PDQ driver spec. I'll use the Canon BJC−210 as the example printer.

First, we look at the database entry for this printer. Note that it is supported "perfectly", so we can expect to get comparable results (or better) to Windows users. The important information is in three places in the entry:

Driver

The last line in the Works?/Language/Driver column tells us one driver that works with this printer. More importantly, this name is a link to the driver's home page.

Notes

The human−readable notes will often contain useful information. For some printers, there is a More Info link, which usually refers to a web page run by a user with this printer, or to the driver's home page.

Driver List

Most printers have a list of driver command data. This is the most important part. A PDQ driver spec has two logical functions: user interaction, and print job processing. These are represented in the file in three places:

Option Declarations

These define what options the user can set, and declare PDQ variables for later parts of the driver to use.

Language Filters

These process the print job from whatever format it arrived in (typically Postscript or ASCII) into a language the printer can understand (for example, PCL). Option values are available here, as well as in the output filter.

Output Filter

This final filter bundles up the printer data regardless of input type; often printer options are set here. Let's work on each of these for a Canon BJC−210:

Options

The driver list for this printer looks like this:

```
 Driver: Ghostscript: −sDEVICE=bj200 −r360x360   # (360x360 BW) Driver: Ghostscript: −sDEV
```

The database's documentation tells us that a "Ghostscript" driver type's text is a set of options for Ghostscript, less the "usual" options like `−q` or the file specifying options.

So, as far as the user is concerned, the BJC−210 supports one useful option: the user should pick color or black−and−white. Let's declare that as choice option called "MODE":

```
 option {   var = "MODE"   desc = "Print Mode"   # default_choice "Color"    # uncomment t
```

the above choice declarations, the user will see a Color or BW choice in the driver options dialog when he prints from xpdq. In the command−line pdq tool, he may specify `−oBW` or `−oColor`. The default can be set from xpdq, or declared above with the `default_choice` keyword.

Language Filtering

PDQ normally identifies its input with the file(1) command. For each type returned by file that you want to handle, you provide a `language_driver` clause. The clause consists mostly of a script to process the printjob language, in any (!) scripting language you wish (the default is the usual Bourne shell).

In our case, we want to print Postscript and ASCII on our BJC−210. This needs two language drivers: one to run Ghostscript for Postscript jobs, and one to add carriage returns to ASCII jobs:

```
 # The first language_driver in the file that matches what file(1)  # says is what gets us
```

That's it! While other printers may need output filtering (as described in the next section), the above clauses are it for the BJC−210. We just wrap them all up in a named `driver` clause:

```
driver canon-bjc210-0.1 {   option {     var = "MODE"     desc = "Print Mode"     # defau
```

Output Filtering

If you want to prepend or append something to all printjobs, or do some sort of transformation on all the data of all types, then it belongs in the `filter_exec` clause. Our little Canon doesn't require such a clause, but just to have an example, here's a simple illustration showing how to support duplexing and resolution choice on a Laserjet or clone that speaks PJL:

```
driver generic-ljet4-with-duplex-0.1 {   # First, two option clauses for the user-selecta
```

7.2 Configuring LPD

Most Linux systems ship with LPD. This section describes a very basic setup for LPD; further sections detail the creation of complex filters and network configuration.

Traditional lpd configuration

The minimal setup for lpd results in a system that can queue files and print them. It will not pay any attention to wether or not your printer will understand them, and will probably not let you produce attractive output. Nevertheless, it is the first step to understanding, so read on!

Basically, to add a print queue to lpd, you must add an entry in */etc/printcap*, and make the new spool directory under */var/spool/lpd*.

An entry in */etc/printcap* looks like:

```
# LOCAL djet500 lp|dj|deskjet:\         :sd=/var/spool/lpd/dj:\         :mx#0:\         :
```

defines a spool called *lp*, *dj*, or *deskjet*, spooled in the directory */var/spool/lpd/dj*, with no per−job maximum size limit, which prints to the device */dev/lp0*, and which does not have a banner page (with the name of the person who printed, etc) added to the front of the print job.

Go now and read the man page for *printcap*.

The above looks very simple, but there a catch − unless I send in files a DeskJet 500 can understand, this DeskJet will print strange things. For example, sending an ordinary Unix text file to a deskjet results in literally interpreted newlines, and gets me:

```
This is line one.                   This is line two.                                   Th
```

nauseam. Printing a PostScript file to this spool would get a beautiful listing of the PostScript commands, printed out with this "staircase effect", but no useful output.

Clearly more is needed, and this is the purpose of filtering. The more observant of you who read the printcap man page might have noticed the spool attributes *if* and *of*. Well, *if*, or the input filter, is just what we need here.

If we write a small shell script called *filter* that adds carriage returns before newlines, the staircasing can be eliminated. So we have to add in an *if* line to our printcap entry above:

```
lp|dj|deskjet:\         :sd=/var/spool/lpd/dj:\         :mx#0:\         :lp=/dev/lp0:\
```

simple filter script might be:

```
#!perl # The above line should really have the whole path to perl # This script must be e
```

we were to do the above, we'd have a spool to which we could print regular Unix text files and get meaningful results. (Yes, there are four million better ways to write this filter, but few so illustrative. You are encouraged to do this more efficiently.)

The only remaining problem is that printing plain text is really not too hot − surely it would be better to be able to print PostScript and other formatted or graphic types of output. Well, yes, it would, and it's easy to do. The method is simply an extention of the above linefeed−fixing filter. If you write a filter than can accept arbitrary file types as input and produce DeskJet−kosher output for each case, then you've got a clever print spooler indeed!

Such a filter is called a *magic* filter. Don't bother writing one yourself unless you print strange things − there are a good many written for you already on the net. APS Filter is among the best, or your Linux distribution may have a printer setup tool that makes this all really easy.

There's one catch to such filters: some older version of lpd don't run the *if* filter for remote printers,

and some do. The version of lpd with modern Linux distributions, and FreeBSD does; most commercial unices that still ship lpd have a version that does not. See the section on network printing later in this document for more information on this.

Accounting

Some installations need to keep track of who prints how much; this section summarizes methods for doing this.

Regular LPD provides very little to help you with accouting. You can specify the name of an accounting file in the af= printcap attribute, but this is merely passed as an argument to your if= filter. It's up to you to make your if= filter write entries to the accounting file, and up to you to process the accounting file later (the traditional format is mainly useful for line printers, and is nontrivial to parse in Perl, so there's no reason to preserve it).

Ghostscript provides a PageCount operator that you can use to count the number of pages in each job; basically you just tack a few lines of postscript onto the end of the job to write an accounting file entry; for the best example of this see the file unix−lpr.sh in the Ghostscript source distribution. Note that the unix−lpr implementation of accounting writes to a file from the Ghostscript interpreter, and is thus incompatible with the recommended −dSAFER option. A better solution might be to query the printer with a PJL command after each job, or to write a postscript snippet that prints the pagecount on stdout, where it can be captured without having to write to a file.

The LPRng print spooler includes an HP−specific sample implementation of accounting; I assume that it queries the printer with PJL.

Large Installations

Large installations, by which I mean networks including more than two printers or hosts, have special needs. Here is a description of one possible arrangement.

Each printer should have a single point of control, where an administrator can pause, reorder, or redirect the queue. To implement this, have everyone printing to a local server, which will then queue jobs and direct them to the proper printer.

Use LPRng, at least on servers; the BSD LPD is too buggy for "real" use.

Client systems should not have unique printing configurations. To implement this, use LPRng's extended printcap syntax so that you have one printcap to use everywhere.

Print queues should not be named for make or model; name print queues for something sensible like location (floor2_nw) or capability (color_transparency). Three years from now, when a printer breaks, you will be able to replace it with a different make or model without causing confusion.

Operate a web page which shows detailed information on each printer, including location, capabilities, etc. Consider having it show the queue. Complex networked environments are unmanagable for users without proper documentation.

On Unix systems, use PDQ to allow selection of print job attributes such as duplex or paper size, and to force users to run all Ghostscript processing under the proper user ID.

On Windows and Apple systems, use either the platform−specific drivers **everywhere** (Samba supports the Windows automagical driver−download mechanism) or use generic Postscript drivers **everywhere**. Do not mix and match; primitive word processors often produce different output when the installed printer driver changes; users cannot deal with output that vaires depending on the particular client/printer pair.

If at all possible, buy a large−volume printer for large−volume printing. If on a budget, use LPRng's multiple printers/one queue facility and assign a babysitter; printers are complex mechanical devices that will often jam and run out of paper in such configurations.

Do not feel that printers must be plugged into workstations; Ethernet "print servers" now cost under $100. The ability to locate printers anywhere you can network is a big improvement over forced location near a host; locate printers in sensible, central locations.

File Permissions

By popular demand, I include below a listing of the permissions on interesting files on my system.

There are a number of better ways to do this, ideally using only SGID binaries and not making everything SUID root, but this is how my system came out of the box, and it works for me. (Quite frankly, if your vendor can't even ship a working lpd you're in for a rough ride).

```
 -r-sr-sr-x   1 root     lp    /usr/bin/lpr* -r-sr-sr-x   1 root     lp    /usr/bin/lprm*
```

Lpd must currently be run as root so that it can bind to the low-numbered lp service port. It should probably become UID lp.lp or something after binding, but I don't think it does. Bummer.

PDQ uses a different, non-daemon-centric scheme, so it has different programs. The only SUID root programs are the lpd interface programs `lpd_cancel`, `lpd_print`, and `lpd_status`; these are SUID because actual Unix print servers require print requests to originate from a priviledged port. If the only printers for which you use PDQ's bsd-lpd interface are networked print servers (like the HP JetDirect or Lexmark's MarkNet adapters) then you do not need the suid bit on these programs.

8.Vendor Solutions

This section is, by definition, incomplete. Feel free to send in details of your favourite distribution. At the moment, I am aware of no distribution that supports, or even provides, the software I recommend: PDQ.

For a while, there were several packages out there all trying to make printer configuration with regular lpd easier. They probably all still exist, but one of the best and most up-to-date is Andreas Klemm's APS Filter package, which has a menu-driven printcap configurator and handles practically any type of input imaginable. If your vendor doesn't ship a nice printer setup tool, APS Filter is another choice; several distributions include apsfilter, or it's an easy add-on.

8.1 Red Hat

Red Hat has a GUI printer administration tool called printtool which can add remote printers and printers on local devices. It lets you choose a ghostscript-supported printer type and Unix device file to print to, then installs a print queue in */etc/printcap* and uses the magic filter program from the rhs-printfilters package to support postscript and other common input types. This solution works fairly well, and is trivial to setup for common cases.

Where Red Hat fails is when you have a printer which isn't supported by their standard Ghostscript (which is GNU rather than Aladdin Ghostscript, and which supports fewer printers). Check in the printer compatibility list above (or online) if you find that you can't print properly with the stock Red Hat software. If your printer isn't supported by Red Hat's tools, you may need to install a contributed verison of Aladdin Ghostscript, and will probably also be better off if you use the apsfilter package, which knows all about the printers supported by late-model Ghostscripts.

In future versions of Red Hat the printtool will be reimplemented to support a larger list of printers and with the intent to support an eventual rhs-printfilters replacement (the current filter has difficultly with many common printers like some non-PCL DeskJets and most Lexmarks). Some VA Linux-developed PPD features may be incorporated, as well.

8.2 Debian

Debian offers a choice between plain lpd and LPRng; LPRng is probably a better choice. I believe Debian also offers a choice of printer configuration tools; apsfilter version 5 or later is probably your best bet, since that verison adds support for LPRng and Ghostscript's uniprint driver scheme.

8.3 SuSE

The printing system on SuSE Linux is based on apsfilter, with some enhancements; SuSE's apsfilter will recognize all common file formats (including HTML, if html2ps is installed). There are two ways to setup printers on SuSE systems:

YaST will let you configure "PostScript", "DeskJet" and "Other printers", supported by Ghostscript drivers; it's also possible to setup HP's GDI printers (DeskJet 710/720, 820, 1000, via the "ppa" package; at this moment b/w only). YaST will provide /etc/printcap entries for every printer ("raw", "ascii", "auto" and "color", if the printer to configure is a color printer). YaST will create spool directories and it will arrange apsfilterrc files, where you're able to fine tune some settings

(Ghostscript preloads, paper size, paper orientation, resolution, printer escape sequences, etc.). With YaST it's also possible to setup network printers (TCP/IP, Samba, or Novell Netware Printer).
In addition there's the regular SETUP program from the original apsfilter package (with some enhancements); run `lprsetup' to invoke this configuration script. Once you get accustomed to its GUI, you'll be able to configure quickly local and network printers (with *local* filtering via the "bypass" feature − that's quite handy).
The SuSE installation manual explains both of these setup procedures.
Wolf Rogner reported some difficulties with SuSE. Apparently the following bugs may bite:
Apsfilter's regular SETUP script is a bit broken, as are the KDE setup tools. Use YaST.
For networked printers that need to be fed from Ghostscript, you'll need to first uncomment the line REMOTE_PRINTER="remote" in /etc/apsfilterrc. Then run YaST to configure the printer and, under Network configurations, set up a remote printer queue.
YaST's setup doesn't allow color laser printers, so configure a mono printer and then change mono to color everwhere in the printcap entry. You may have to rename the spool directory, too.

8.4 Other Distributions

Please send me info on what other distributions do!

9.Ghostscript.

Ghostscript is an incredibly significant program for Linux printing. Most printing software under Unix generates PostScript, which is typically a $100 option on a printer. Ghostscript, however, is free, and will generate the language of your printer from PostScript. When tied in with your PDQ printer driver declaration or `lpd` input filter, it gives you a virtual PostScript printer and simplifies life immensely.
Ghostscript is available in two forms. The commercial version of Ghostscript, called Aladdin Ghostscript, may be used freely for personal use but may not be distributed by commercial Linux distributions. It is generally a year or so ahead of the free Ghostscript; at the moment, for example, it supports many color inkjets that the older Ghostscripts do not.
The free version of Ghostscript is GNU Ghostscript, and is simply an aged version of Aladdin ghostscript kindly given to GNU. (Kudos to Aladdin for this arrangement; more software vendors should support free software in this way, if they can't handle full−blown GPL distribution of their code).
Whatever you do with `gs`, be very sure to run it with the option for disabling file access (`-dSAFER`). PostScript is a fully functional language, and a bad PostScript program could give you quite a headache.
Speaking of PDF, Adobe's Portable Document Format is actually little more than organized PostScript in a compressed file. Ghostscript can handle PDF input just as it does PostScript. So you can be the first on your block with a PDF−capable printer.

9.1 Invoking Ghostscript

Typically, Ghostscript will be run by whatever magic filter you settle upon (I recommend apsfilter if your vendor didn't supply anything that suits you), but for debugging purposes it's often handy to run it directly.
`gs -help` will give a brief informative listing of options and available drivers (note that this list is the list of drivers compiled in, not the master list of all available drivers).
You might run gs for testing purposes like: `gs` *`options`* `-q -dSAFER -sOutputFile=/dev/lp1 test.ps`.

9.2 Ghostscript output tuning

There are a number of things one can do if gs's output is not satisfactory (actually, you can do anything you darn well please, since you have the source).
Some of these options, and others described in the Ghostscript User Guide (the file `Use.htm` in the Ghostscript distribution; possibly installed under /usr/doc or /usr/share/doc on your system) are all

excellent candidates for driver options in your PDQ driver declaration.

Output location and size

The location, size, and aspect ratio of the image on a page is controlled by the printer−specific driver in ghostscript. If you find that your pages are coming out scrunched too short, or too long, or too big by a factor of two, you might want to look in your driver's source module and adjust whatever parameters jump out at you. Unfortunately, each driver is different, so I can't really tell you what to adjust, but most of them are reasonably well commented.

Gamma, dotsizes, etc.

Most non−laser printers suffer from the fact that their dots are rather large. This results in pictures coming out too dark. If you experience this problem you should use your own transfer function. Simply create the following file in the ghostscript lib−dir and add its name to the gs call just before the actual file. You may need to tweak the actual values to fit your printer. Lower values result in a brighter print. Especially if your driver uses a Floyd−Steinberg algorithm to rasterize colors, lower values (0.2 − 0.15) are probably a good choice.

```
 ---8<---- gamma.ps ----8<--- %! %transfer functions for cyan magenta yellow black {0.3 ex
```

It is also possible to mend printers that have some kind of colour fault by tweaking these values. If you do that kind of thing, I recommend using the file colorcir.ps, that comes with ghostscript (in the examples/ subdir), as a test page.

For many of the newer color inkjet drivers, there are command−line options, or different upp driver files, which implement gamma and other changes to adapt the printer to different paper types. You sould look into this before playing with Postscript to fix things.

Color Printing in Ghostscript

Ghostscript's default color dithering is optimized for low−resolution devices. It will dither rather coarsely in an attempt to produce 60ppi output (not dpi, ppi − the "apparent" color pixels per inch you get after dithering). This produces rather poor output on modern color printers; inkjets with photo paper, in particular, are capable of mich finer ppi settings.

To adjust this, use the Ghostscript option −dDITHERPPI=x, where x is the value to use. This may or may not have an effect with all drivers; many newer drivers implement their own dithering and pay no attention to this setting. Some drivers can use either the regular Ghostscript or driver−specific dithering.

This makes for an excellent argument in a PDQ driver declaration, if it applies.

10.How to print to a printer over the network

One of the features of pdq and lpd is that they support printing over the network to printers physically connected to a different machine. With the careful combination of filter scripts and assorted utilities, you can make either print transparently to printers on all sorts of networks.

10.1 To a Unix/lpd host

To allow remote machines to print to your printer using the LPD protocol, you must list the machines in */etc/hosts.equiv* or */etc/hosts.lpd*. (Note that *hosts.equiv* has a host of other effects; be sure you know what you are doing if you list any machine there). You can allow only certain users on the other machines to print to your printer by usign the *rs* attribute; read the `lpd` man page for information on this.

With pdq

With PDQ, you define a printer with the interface type "bsd−lpd". This interface takes arguments for the remote hostname and queue name; the printer definition wizard will prompt you for these.

With `lpd`

To print to another machine, you make an */etc/printcap* entry like this:

```
 # REMOTE djet500 lp|dj|deskjet:\         :sd=/var/spool/lpd/dj:\         :rm=machine.out.
```

that there is still a spool directory on the local machine managed by `lpd`. If the remote machine is

busy or offline, print jobs from the local machine wait in the spool area until they can be sent.

With `rlpr`

You can also use *rlpr* to send a print job directly to a queue on a remote machine without going through the hassle of configuring lpd to handle it. This is mostly useful in situations where you print to a variety of printers only occasionally. From the announcement for *rlpr*:

Rlpr uses TCP/IP to send print jobs to lpd servers anywhere on a network.

Unlike lpr, it *does not* require that the remote printers be explicitly known to the machine you wish to print from, (e.g. through */etc/printcap*) and thus is considerably more flexible and requires less administration.

rlpr can be used anywhere a traditional lpr might be used, and is backwards compatible with traditional BSD lpr.

The main power gained by rlpr is the power to print remotely *from anywhere to anywhere* without regard for how the system you wish to print from was configured. Rlpr can work as a filter just like traditional lpr so that clients executing on a remote machine like netscape, xemacs, etc, etc can print to your local machine with little effort.

Rlpr is available from `Metalab`.

10.2 To a Win95, WinNT, LanManager, or Samba printer

There is a Printing to Windows mini−HOWTO out there which has more info than there is here.

From PDQ

There is not a prebuilt smb interface that I am aware of, but it would be fairly easy to create using the model set by the Netatalk−based appletalk interface. Someone please create one and submit it for inclusion!

Read the Windows/LPD section below for more tips on how to do it.

From LPD

It is possible to direct a print queue through the `smbclient` program (part of the samba suite) to a TCP/IP based SMB print service. Samba includes a script to do this called `smbprint`. In short, you put a configuration file for the specific printer in question in the spool directory, and install the `smbprint` script as the *if*.

The */etc/printcap* entry goes like this:

```
lp|remote−smbprinter:\     :lp=/dev/null:sh:\     :sd=/var/spool/lpd/lp:\     :if=/usr/lo
```

You should read the documentation inside the `smbprint` script for more information on how to set this up.

You can also use `smbclient` to submit a file directly to an SMB printing service without involving `lpd`. See the man page.

10.3 To a NetWare Printer

The ncpfs suite includes a utility called `nprint` which provides the same functionality as `smbprint` but for NetWare. You can get ncpfs from Metalab. From the LSM entry for version 0.16:

With ncpfs you can mount volumes of your netware server under Linux. You can also print to netware print queues and spool netware print queues to the Linux printing system. You need kernel 1.2.x or 1.3.54 and above. ncpfs does NOT work with any 1.3.x kernel below 1.3.54.

From LPD

To make `nprint` work via lpd, you write a little shell script to print stdin on the NetWare printer, and install that as the *if* for an lpd print queue. You'll get something like:

```
sub2|remote−NWprinter:\         :lp=/dev/null:sh:\         :sd=/var/spool/lpd/sub2:\
```

`nprint−script` might look approximately like:

```
#! /bin/sh # You should try the guest account with no password first! /usr/local/bin/npri
```

10.4 To an EtherTalk (Apple) printer

The netatalk package includes something like `nprint` and `smbclient`. Others have documented

the procedure for printing to and from an Apple network far better than I ever will; see the Linux Netatalk−HOWTO.

From PDQ

PDQ includes an interface declaration called "appletalk". This uses the Netatalk package to print to a networked Apple printer. Just select this interface in xpdq's "Add printer" wizard.

10.5 To an HP or other ethernet printer

HPs and some other printers come with an ethernet interface which you can print to directly using the lpd protocol. You should follow the instructions that came with your printer or its network adaptor, but in general, such printers are "running" lpd, and provide one or more queues which you can print to. An HP, for example, might work with a printcap like:

```
lj−5|remote−hplj:\          :lp=/dev/null:sh:\          :sd=/var/spool/lpd/lj−5:\          :
```

using the PDQ bsd−lpd interface arguments of REMOTE_HOST=printer.name.com and QUEUE=raw.

HP Laserjet printers with Jet Direct interfaces generally support two built in lpd queues − "raw" which accepts PCL (and possibly Postscript) and "text" which accepts straight ascii (and copes automatically with the staircase effect). If you've got a JetDirect Plus3 three−port box, the queues are named "raw1", "text2", and so forth.

Note that the ISS company has identified an assortment of denial of service attacks which hang HP Jetdirect interfaces. Most of these have been addressed beginning in Fall 98.

In a large scale environment, especially a large environment where some printers do not support PostScript, it may be useful to establish a dedicated print server to which all machines print and on which all ghostscript jobs are run. This will allow the queue to be paused or reordered using the topq and lprm commands.

This also allows your Linux box to act as a spool server for the printer so that your network users can complete their print jobs quickly and get on with things without waiting for the printer to print any other job that someone else has sent. This is suggested too if you have unfixable older HP Jetdirects; it reduces the likelihood of the printers wedging.

To do this, set up a queue on your linux box that points at the ethernet equipped HP LJ (as above). Now set up all the clients on your LAN to point at the Linux queue (eg lj−5 in the example above).

Some HP network printers apparently don't heed the banner page setting sent by clients; you can turn off their internally generated banner page by telnetting to the printer, hitting return twice, typing "banner: 0" followed by "quit". There are other settings you can change this way, as well; type "?" to see a list.

The full range of settings can be controlled with HP's WebJet software. This package runs as a daemon, and accepts http requests on a designated port. It serves up forms and Java applets which can control HP printers on the network. In theory, it can also control Unix print queues, but it does so using the rexec service, which is completely unsecure. I don't advise using that feature.

To older HPs

Some printers (and printer networking "black boxes") support only a cheesy little non−protocol involving plain TCP connections. Notable in this category are early−model JetDirect (including some JetDirectEx) cards. Basically, to print to the printer, you must open a TCP connection to the printer on a specified port (typically 9100, or 9100, 9101 and 9102 for three−port boxes) and stuff your print job into it. LPRng has built−in support for stuffing print jobs into random TCP ports, but with BSD lpd it's not so easy. The best thing is probably to obtain and use the little utility called netcat.

A netcat−using PDQ interface would look something like this:

```
interface tcp−port−0.1 {     help "This is one of the first interfaces supported by standa
```

Failing that, it can be implemented, among other ways, in Perl using the program below. Or, or better performance, use the program netcat ("nc"), which does much the same thing in a general purpose way. Most distributions should have netcat available in prepackaged form.

```
#!/usr/bin/perl # Thanks to Dan McLaughlin for writing the original version of this # scri
```

10.6 Running an *if* for remote printers with old LPDs

One oddity of older version of lpd is that the *if* is not run for remote printers. (Version after 0.43 or so have the change originated on FreeBSD such that the if is always run). If you find that you need to run an *if* for a remote printer, and it isn't working with your lpr, you can do so by setting up a double queue and requeueing the job. As an example, consider this *printcap*:

```
lj-5:\         :lp=/dev/null:sh:\         :sd=/var/spool/lpd/lj-5:\         :if=/usr/lib/
```

light of this `filter-lj-5` script:

```
#!/bin/sh gs <options> -q -dSAFER -sOutputFile=- - | \         lpr -Plj-5-remote -U$5
```

The *-U* option to lpr only works if lpr is run as daemon, and it sets the submitter's name for the job in the resubmitted queue correctly. You should probably use a more robust method of getting the username, since in some cases it is not argument 5. See the man page for *printcap*.

10.7 From Windows.

Printing from a Windows (or presumably, OS/2) client to a Linux server is directly supported over SMB through the use of the SAMBA package, which also supports file sharing of your Linux filesystem to Windows clients.

Samba includes fairly complete documentation, and there is a good Samba FAQ which covers it, too. You can either configure a magic filter on the Linux box and print PostScript to it, or run around installing printer-specific drivers on all the Windows machines and having a queue for them with no filters at all. Relying on the Windows drivers may in some cases produce better output, but is a bit more of an administrative hassle if there are many Windows boxen. So try Postscript first.

With PDQ, you should configure Samba to run the pdq command with appropriate arguments instead of the lpr command that it defaults to running. I believe that Samba will run pdq as the proper user, so it should work well this way. There are several Samba options that you should adjust to do this:

printcap

This should point to a "fake" printcap you whip up listing available printers. All you need is a short and long name for each printer, one per line:

```
lp1|Printer One lp2|Printer Two lp3|Printer Three
```

The short name will be used as the printer name for the print command:

print command

This will need to be set to something like `pdq -P %p %s ; rm %s`.

lprm command

There doesn't seem to be a good value for this setting at the moment. PDQ's queued jobs will expire after a time, so if the printer is totally gone there's no problem. If you just change your mind, you can use xpdq to cancel jobs, but this is inconvenient from Windows. Just put a do-nothing command like true for now. If you use lpd or lprng as the back-end, then a suitable lprm command should work. I'm not sure how Samba would identify the lpr queue entry number for a pdq-submitted job.

lpq command

Again, PDQ doesn't offer a good value to put here. Distributed systems don't offer a sensible way to see the queue, but samba-centric centralized server systems to have a queue worth examining. Just put a do-nothing command like true for now. If you use lpd or lprng as the back-end, then a suitable lpq command should work; you just won't see jobs until they're done being filtered by PDQ.

10.8 From an Apple.

Netatalk supports printing from Apple clients over EtherTalk. See the Netatalk HOWTO Page for more information.

10.9 From Netware.

The ncpfs package includes a daemon named pserver which can be used to privide service to a NetWare print queue. From what I understand, this system requires a Bindery−based NetWare, ie 2.x, 3.x, or 4.x with bindery access enabled.

For more information on ncpfs and it's pserver program, see the ncpfs FTP site.

11.Windows−only printers

As I discussed earlier, some printers are inherently unsupported because they don't speak a normal printer language, instead using the computer's CPU to render a bitmap which is then piped to the printer at a fixed speed. In a few cases, these printers also speak something normal like PCL, but often they do not. In some (really low−end) cases, the printer doesn't even use a normal parallel connection but relies on the vendor's driver to emulate what should be hardware behaviour (most importantly flow control).

In any case, there are a few possible workarounds if you find yourself stuck with such a lemon.

11.1 The Ghostscript Windows redirector

There is now a Windows printer driver available (called mswinpr2) that will run a print job through Ghostscript before finally printing it. (Rather like an if filter in Unix's LPD). There is also a new Ghostscript driver which will print using Windows GDI calls. Taken all together, this allows a Windows machine to print PostScript to a Windows−only printer through the vendor's driver.

If you get that working, you can then follow the instructions above for printing to a Windows printer over the network from Linux to let Unix (and other Windows, Mac, etc) hosts print to your lemon printer.

11.2 HP Winprinters

Some HP printers use "Printing Performance Architecture" (marketingspeak for "we were too cheap to implement PCL"). This is supported in a roundabout way via the pbm2ppa translator written by Tim Norman. Basically, you use ghostscript to render PostScript into a bitmapped image in pbm format and then use pbm2ppa to translate this into a printer−specific ppa format bitmap ready to be dumped to the printer. This program may also come in ghostscript driver format by now.

The ppa software can be had from the ppa home page; pbm2ppa supports some models of the HP 720, 820, and 1000; read the documentation that comes with the package for more details on ppa printer support.

11.3 Lexmark Winprinters

Most of the cheap Lexmark inkjets use a proprietary language and are therefore Winprinters. However, Henryk Paluch has written a program which can print on a Lexmark 7000. Hopefully he'll be able to figure out color and expand support to other Lexmark inkjets. See here for more info. Similarly, there are now drivers for the 5700, 1000, 1100, 2070, and others. See the supported printers listing above, and my web site, for more information on obtaining these drivers.

12.How to print to a fax machine.

You can print to a fax machine with, or without, a modem.

12.1 Using a faxmodem

There are a number of fax programs out there that will let you fax and receive documents. One of the most complex is Sam Leffler's *HylaFax*, available from `ftp.sgi.com`. It supports all sorts of things from multiple modems to broadcasting.

SuSE ships a Java HylaFax client which allegedly works on any Java platform (including Windows

and Linux). There are also non−Java fax clients for most platforms; Linux can almost certainly handle your network faxing needs.
Also available, and a better choice for most Linux boxen, is `efax`, a simple program which sends faxes. The getty program `mgetty` can receive faxes (and even do voicemail on some modems!).

Faxing from PDQ

PDQ doesn't ship with a fax interface declaration, but here's a simple one (which is only partly tested):

```
interface efax−0.1 {    help "This interface uses the efax package's fax program to send a
```

12.2 Using the Remote Printing Service

There is an experimental service offered that lets you send an email message containing something you'd like printed such that it will appear on a fax machine elsewhere. Nice formats like postscript are supported, so even though global coverage is spotty, this can still be a very useful service. For more information on printing via the remote printing service, see the Remote Printing WWW Site.

13. How to generate something worth printing.

Here we get into a real rat's−nest of software. Basically, Linux can run many types of binaries with varying degrees of success: Linux/x86, Linux/Alpha, Linux/Sparc, Linux/foo, iBCS, Win16/Win32s (with dosemu and, someday, with Wine), Mac/68k (with Executor), and Java. I'll just discuss native Linux and common Unix software.
For Linux itself, choices are mostly limited to those available for Unix in general:

13.1 Markup languages

Most markup languages are more suitable for large or repetitive projects, where you want the computer to control the layout of the text to make things uniform.

nroff

This was one of the first Unix markup languages. Man pages are the most common examples of things formatted in *roff macros; many people swear by them, but nroff has, to me at least, a more arcane syntax than needed, and probably makes a poor choice for new works. It is worth knowing, though, that you can typeset a man page directly into postscript with groff. Most man commands will do this for you with `man −t foo | lpr`.

TeX

TeX, and the macro package LaTeX, are one of the most widely used markup languages on Unix. Technical works are frequently written in LaTeX because it greatly simplifies the layout issues and is *still* one of the few text processing systems to support mathematics both completely and well. TeX's output format is `dvi`, and is converted to PostScript or Hewlett Packard's PCL with `dvips` or `dvilj`. If you wish to install TeX or LaTeX, install the whole teTeX group of packages; it contains everything. Recent TeX installations include pdfTeX and pdfLaTeX, which produce Adobe PDF files directly. Commands are available do create hyperlinks and navigation features in the PDF file.

SGML

There is at least one free sgml parser available for Unix and Linux; it forms the basis of Linuxdoc−SGML's homegrown document system. It can support other DTD's, as well, most notable DocBook

HTML

Someone suggested that for simple projects, it may suffice to write it in HTML and print it out using Netscape. I disagree, but YMMV.

13.2 WYSIWYG Word Processors

There is no longer any shortage of WYSIWYG word processing software. Several complete office suites are available, including one that's free for personal use (StarOffice).

StarOffice

Sun Microsystems is distributing StarOffice on the net free for Linux. This full−blown office suite has all the features you'd expect, and you can't beat the price. There's a mini−HOWTO out there which describes how to obtain and install it. It generates PostScript or PCL, so should work with most any printer that works otherwise on Linux. Apparently it's an Office clone and is rather bloated; these are probably two equivalent facts!

WordPerfect

Corel distributes a basic version of Word Perfect 8 free for Linux, and has suggested that they will distribute Corel Draw and Quattro Pro as well, once they are ported. This is probably the best option if you have an ARM machine; Corel makes the ARM−based Netwinder Linux computers and is almost certian to offer ARM Linux versions of everything. You can also buy the full−blown version and support, together or separately. The Linux WordPerfect Fonts and Printers page has information about configuring WordPerfect for use with either Ghostscript or its built−in printer drivers (which are apparently identical the DOS WordPerfect drivers, if your printer's driver isn't included in the WP8 distribution).

Applix

Applix is a cross−platform (ie, various Unices, Windows, and others) office suite sold by the Applix company. Red Hat and SuSE sold it themselves when it was the only game in town; now sales have reverted to Applix.

AbiWord

AbiWord is one of several GPL WYSIWYG word processor projects; this one has produced a very nice word processor based on an XML format and capable of Word file import.

LyX

LyX is a front−end to LaTeX which looks very promising. See the LyX Homepage for more information. There is a KDE−styled version of LyX, called Klyx; the author of LyX and the instigator of KDE are the same person.

Maxwell

Maxwell is a simple MS RTF−format based word processor which started as a commercial product but is now distributed under the GPL.

The Andrew User Interface System

AUIS includes `ez`, a WYSIWYG−style editor with most basic word processor features, HTML

capabilities, and full MIME email and newsgroup support. Unfortunately, AUIS is no longer maintained.

Koffice

The KDE project is working toward a whole office suite. I don't think it's ready for prime time yet. The word processor will apparently be a descendant of LyX.

GNOME

The GNOME project also is working toward various GNU−licensed officey tools. None are available yet, though.
Other vendors should feel free to drop me a line with your offerings.

13.3 Printing Photographs

There are many details to getting decent photo output from common printers.

Ghostscript and Photos

Ghostscript has some difficulties rendering color photographs through most drivers. The problems are several:
Many drivers have poorly tuned color support. Often the colors don't match the Windows driver output or the screen. OTOH, all drivers, and Ghostscript as a whole, have readily adjustable color support; the "Gamma" settings are one thing to play with, and there are others documented in Ghostscript's Use.htm documentation file.
I'm only aware of one Ghostscript driver with support for 6 and 7 color printing; it's in beta at the moment and supports a few Epson Stylus Photo models. It is rumoured to produce better color than the Windows driver (!).
Ghostscript often ends up dithering coarsely, or generating printouts with artifacts like banding. The dithering can usually be corrected; see the color in ghostscript section above, and read the documentation for your driver. You should be able to correct some of these problems by tuning Ghostscript; see the Ghostscript section above for more information on how to do this. Fiddling with Ghostscript options is much easier if you declare them as options in a PDQ driver declaration.
That said, the obvious solution is to use non−Ghostscript software for printing photos, and indeed, such things do exist. The main contender is the print plugin in the Gimp, which supports pixel−for−pixel printing on Epson Styluses and Postscript printers (with basic PPD support). That driver will shortly be available for Ghostcript, as well. Also possible to use for this purpose are the assorted external pnm−to−foo programs used to print on printers like the cheap Lexmarks; these print pixmaps pixel−for−pixel. A print−via−filter option shouldn't be too hard to add to the Gimp.
The best solution, of course, is to buy a Postscript printer; such printers can usually be completely controlled from available free software, and will print to the full capability of the printer.

Paper

Color inkjets are extremely dependant on the paper for good output. The expensive glossy coated inkjet papers will allow you to produce near−photographic output, while plain uncoated paper will often produce muddy colors and fuzzy details. Nonglossy coated inkjet papers will produce results in between, and are probably best for final prints of text, as well. Stiffer glossy coated "photo" papers will produce similar output to lighter−weight glossy papers, but will feel like a regular photo.

Printer Settings

For photo output on most color inkjets, you should use the most highly interlaced (and slowest) print mode; otherwise solid regions may have banding or weak colors. Generally with Ghostscript this is what will happen when you pick the highest resolution. With Postscript printers, you may need to add a snippet to the prologue based on the settings available in the PPD file. The Gimp's PPD support

doesn't include the print quality setting, but I added it in an ugly way for my own use; contact me if you'd like that. If you use PDQ, you can easily add all the printer settings you need in the driver declaration file; for PJL printers this is particularly easy, and for Postscript printers my ppdtopdq utility can help.

Print Durability

Color inkjet printouts usually fade after a few years, especially if exposed to lots of light and air; this is a function of the ink. Printers with ink−only consumables like the Epsons and Canons can buy archival inks, which are less prone to this problem.

Shareware and Commercial Software

There's a program called xwtools which supports photo printing with all the bells and whistles on an assortment of Epson, HP, and Canon pritners. Unfortunately, it was written under NDA, so comes without source. Unless you use it for the Epson Stylus Color 300 on Linux x86, it costs 15 euros for personal use; commercial pricing is unknown.

The ESP Print Pro package from Easy Software supports many printers which might otherwise be unsupported. Unfortunately, since it is based on Ghostscript 4.03, I don't expect wonderful results with this software for photos. But someone should try.

14.On−screen previewing of printable things.

Nearly anything you can print can be viewed on the screen, too.

14.1 PostScript

Ghostscript has an X11 driver best used under the management of the PostScript previewer gv. The latest versions of these programs should be able to view PDF files, as well. Note that gv has replaced the older previewer "Ghostview"; the new user interface is mch prettier and featureful that ghostview's plain old Athena gui.

14.2 TeX dvi

TeX DeVice Independent files may be previewed under X11 with xdvi. Modern versions of xdvi call ghostscript to render PostScript specials.

A VT100 driver exists as well. It's called `dgvt`. `Tmview` works with Linux and svgalib, if that's all you can do.

14.3 Adobe PDF

Adobe's Acrobat Reader is available for Linux; just download it from their web site http://www.adobe.com/.

You can also use xpdf, which is freeware and comes with source, and I should think Ghostview supports viewing PDF files with gs under X11 by now.

15.Serial printers under lpd

Serial printers are rather tricky under lpd.

15.1 Setting up in printcap

Lpd provides five attributes which you can set in */etc/printcap* to control all the settings of the serial port a printer is on. Read the *printcap* man page and note the meanings of *br#*, *fc#*, *xc#*, *fs#* and *xs#*. The last four of these attributes are bitmaps indicating the settings for use the port. The *br#* atrribute is simply the baud rate, ie `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
```

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.

You actually use stty in a somewhat odd way. Since stty operates on the terminal connected to it's standard input, you use it to manipulate a given serial port by using the `<' character as above.
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/asm−i386/termbits.h. This contains a lot of #defines 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). Go to the section that starts out
`/* 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'.

15.2 Older serial printers that drop characters

Jon Luckey points out that some older serial printers with ten−cent serial interfaces and small buffers *really* mean stop when they say so with flow control. He found that disabling the FIFO in his Linux box's 16550 serial port with `setserial` corrected the problem of dropped characters (you apparently just specify the uart type as an 8250 to do this).

16. Credits

Special thanks to Jacob Langford, author of `pdq`, who finally gave us something better than the smattering of scripts globbed onto a 20 year old overgrown line−printer control program.
The `smbprint` information is from an article by Marcel Roelofs `<marcel@paragon.nl>`.
The `nprint` information for using Netware printers was provided by Michael Smith `<mikes@bioch.ox.ac.uk>`.
The serial printers under lpd section is from Andrew Tefft `<teffta@engr.dnet.ge.com>`.
The blurb about gammas and such for gs was sent in by Andreas `<quasi@hub−fue.franken.de>`.
The two paragraphs about the 30 second closing_wait of the serial driver was contributed by Chris Johnson `<cdj@netcom.com>`.
Robert Hart sent a few excellent paragraphs about setting up a print server to networked HPs which I used verbatim.
And special thanks to the dozens upon dozens of you who've pointed out typos, bad urls, and errors in the document over the years.