The Linux Printing HOWTO

Grant Taylor <gtaylor+pht@picante.com>

v3.29, 9 March 1999

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).

Contents

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-dependant 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 9 or 10 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* `<http://www.picante.com/~gtaylor/pht/>` and b) drop me a note saying what ought to be here but isn't.

The *Printing HOWTO Home Page* `<http://www.picante.com/~gtaylor/pht/>` 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 third generation, which is to say the third complete rewrite, 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 (!).

1.2 Copyright

This document is Copyright (c) 1992-1999 by Grant Taylor. Feel free to copy and redistribute this document according to the terms of the GNU General Public License, revision 2 or later.

2 How to print

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* <http://metalab.unc.edu/LDP/HOWTO/Printing-Usage-HOWTO.html> covers this, and a few other queue manipulation commands you should probably know.

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. 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.

I'll cover the parport driver more completely when I find myself using one, but in the meantime you can read the file *Documentation/parport.txt* in your kernel sources, or look at the *parport web site* `<http://www.cyberelk.demon.co.uk/parport.html>`.

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* `<http://metalab.unc.edu/mdw/HOWTO/Serial-HOWTO.html>`.

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-independant

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 independantly 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. The *Ghostscript Home Page* `<http://www.cs.wisc.edu/~ghost/>` has a list of supported printers and information on the status of new and experimental drivers. Note that this page lists supported printers in the latest version of Ghostscript, while 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* `<http://www.picante.com/~gtaylor/pht/printer_list.cgi>` 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* `<http://www.cs.wisc.edu/~ghost/printer.html>` has a list of some working printers, as well as links to other pages. And 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* `<http://www.picante.com/~gtaylor/pht/printer_list.cgi>`, so that it will be listed properly in this document in the future.

4.3.1 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!

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. It should, however, be easy to corroborate from the driver web pages and manufacturer web sites.

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

Brother

Perfectly

HL-10V, HL-660, HL-720, HL-760.

Mostly

HL-1050.

Canon

Perfectly

BJ-10e, BJ-20, BJ-200, BJC-210, BJC-240, BJC-250, BJC-610, BJC-620, BJC-70, BJC-800, LBP-8II, LIPS-III.

Mostly

BJ-300, BJC-4000, BJC-4100, BJC-4200, BJC-4300, BJC-4400, BJC-7000, BJC-7004.

Partially

BJC-4550, MultiPASS C2500, MultiPASS C3500.

Paperweight

BJC-5000, LBP-460, LBP-660.

Epson

Perfectly

ActionLaser 1100, LP 8000, LQ 850, Stylus Color, Stylus Color 400, Stylus Color 500, Stylus Color 600, Stylus Color 640, Stylus Color 850, Stylus Color II, Stylus Color IIs, Stylus Pro XL.

Mostly

Stylus Color 800, Stylus Photo 750.

Partially

Stylus Color 740.

HP

Perfectly

2000Cse, 2500C, DesignJet 650C, DeskJet 1200C, DeskJet 1600C, DeskJet 1600Cm, DeskJet 400, DeskJet 420C, DeskJet 500, DeskJet 550C, DeskJet 600, DeskJet 660Cse, DeskJet 690C, DeskJet 850C, DeskJet 855C, DeskJet 870, DeskJet 870Cxi, DeskJet 890, LaserJet 1100, LaserJet 2100, LaserJet 2100M, LaserJet 4000N, LaserJet 4L, LaserJet 5, LaserJet 5L, LaserJet 5MP, LaserJet 6L, LaserJet 6MP, LaserJet 8000, LaserJet 8100, Laserjet 5000, Mopier 320, PaintJet XL300.

Mostly

HP 660C.

Partially

DeskJet 1000C, DeskJet 670C, DeskJet 710, DeskJet 720C, DeskJet 722C, DeskJet 820C.

Paperweight

LaserJet 3100.

IBM

Perfectly

Jetprinter 3852.

Lexmark

Perfectly

Optra Color 1200, Optra Color 1275, Optra Color 40, Optra Color 45, Optra E, Optra E+, Optra Ep, Optra S 1250.

Partially

1020 Business, 3000, 5700, 7000.

Paperweight

1000, 1020, 1100, 2030, 2050, 2070, 5000, 7200, Winwriter 100, Winwriter 150c, Winwriter 200.

Minolta

Perfectly

PagePro 6, PagePro 8.

NEC

Perfectly

P2X.

Partially

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

Paperweight

SuperScript 660i, SuperScript 660plus.

Okidata

Perfectly

OL 410e, OL 610e/PS, OL 810e/PS, Okipage 6e, Okipage 6ex, Okipage 8c.

Mostly

Okipage 4w.

Paperweight

Okipage 8w, okijet 2010.

Olivetti

Perfectly

JP350S.

Panasonic

Perfectly
KX-P1123, KX-P4440, KX-P5400, KX-P8420, KX-P8475.

Partially
KX-P6500.

Paperweight
KX-P6100, KX-P8410.

QMS

Perfectly
2425 Turbo EX.

Ricoh

Perfectly
4801, 6000.

Xerox

Perfectly
DocuPrint C55, DocuPrint N17, DocuPrint N32.

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. And there are signs that some Linux vendors will shift to providing LPRng, a far less ancient print spooling implementation that is 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.

For the moment, even in light of the new options, lpd is probably fine for most Linux users. While it isn't the snazziest system, it works fine once set up, and it is well understood and extensively documented in third-party Unix books.

If you'd like more information on LPRng, check out the *LPRng Web Page* `<http://www.astart.com/lprng/LPRng.html>`. Future versions of this HOWTO will include information on using both LPRng and regular lpd.

6 How it works, basic

In order to get printing working well, you need to understand how the lpd system works.

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, basic

7.1 Traditional lpd configuration

The minimal setup for lpd rsults 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:\
        :lp=/dev/lp0:\
        :sh:
```

This 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.
                                  This is line three.
```

ad 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:\
        :if=/var/spool/lpd/dj/filter:\
        :sh:
```

A simple filter script might be:

```
#!perl
# The above line should really have the whole path to perl
# This script must be executable: chmod 755 filter
while(<STDIN>){chop $_; print "$_\r\n";};
# You might also want to end with a form feed: print "\f";
```

If 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.

7.2 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*
-rwxr--r--   1 root     root   /usr/sbin/lpd*
-r-xr-sr-x   1 root     lp     /usr/sbin/lpc*
drwxrwxr-x   4 root     lp     /var/spool/lpd/
drwxr-xr-x   2 root     lp     /var/spool/lpd/lp/
```

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.

8 Getting Printing Software

Many prewritten filter packages (and other printer-related software) are available from *Metalab* <ftp://metalab.unc.edu/pub/Linux/system/printing/>. Such utilities as psutils, a2ps, mpage, dvitodvi, flpr, etc can all be found there.

For a while, there were several packages out there all trying to make printer configuration 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 the way to go.

9 Vendor Solutions

This section is, by definition, incomplete. Feel free to send in details of your favourite distribution.

9.1 Red Hat

Red Hat has a GUI printer administration tool (in the control panel) 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 writes a short PostScript-and-ascii magic filter based around `gs` and *nenscript*. 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* <http://www.picante.com/~gtaylor/pht/printer_list.cgi>) 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.

9.2 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.

9.3 Other Distributions

Please send me info on what other distributions do!

10 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 `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.

10.1 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`.

10.2 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).

10.2.1 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.

10.2.2 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 exp} {0.3 exp} {0.3 exp} {0.3 exp} setcolortransfer
---8<------------------8<---
```

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.

11 How to print to a printer over the network

One of the features of lpd is that it supports 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 lpr print transparently to printers on all sorts of networks.

11.1 To a Unix/lpd host

To allow remote machines to print to your printer, 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.

11.1.1 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.there.com:\
        :rp=printername:\
        :lp=/dev/null:\
        :sh:
```

Note 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.

11.1.2 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* <ftp://metalab.unc.edu/pub/Linux/system/printing/>.

11.2 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.

It is possible to direct an lpd 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/local/sbin/smbprint:
```

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.

11.3 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.

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:\
        :if=/var/spool/lpd/nprint-script:
```

The `nprint-script` might look approximately like:

```
#! /bin/sh
# You should try the guest account with no password first!
/usr/local/bin/nprint -S net -U name -P passwd -q printq-name -
```

11.4 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* <http://thehamptons.com/anders/netatalk/>.

11.5 To an HP or other ethernet printer

HPs and some other printers come with an ethernet interface which you can print to directly using lpd. 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:\
        :rm=printer.name.com:rp=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 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* <http://www.hp.com/go/webjetadmin> 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.

11.5.1 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 (which you really should be using instead, if you're reading this section) has built-in support for stuffing print jobs into random TCP ports, but with BSD lpd it's not so easy. This can be implemented, among other ways, in Perl:

```
#!/usr/bin/perl
# Thanks to Dan McLaughlin for writing the original version of this
# script (And to Jim W. Jones for sitting next to Dan when writing me
# for help ;)

$fileName = @ARGV[0];

open(IN,"$fileName") || die "Can't open file $fileName";

$dpi300     = "\x1B*t300R";
$dosCr      = "\x1B&k3G";
$ends = "\x0A";

$port =  9100 unless $port;
$them = "bach.sr.hp.com" unless $them;

$AF_INET = 2;
$SOCK_STREAM = 1;
$SIG{'INT'} = 'dokill';
$sockaddr = 'S n a4 x8';

chop($hostname = `hostname`);
($name,$aliases,$proto) = getprotobyname('tcp');
($name,$aliases,$port) = getservbyname($port,'tcp')
    unless $port =~ /^\d+$/;;
($name,$aliases,$type,$len,$thisaddr) =
        gethostbyname($hostname);
($name,$aliases,$type,$len,$thataddr) = gethostbyname($them);
$this = pack($sockaddr, $AF_INET, 0, $thisaddr);
$that = pack($sockaddr, $AF_INET, $port, $thataddr);

if (socket(S, $AF_INET, $SOCK_STREAM, $proto)) {
#    print "socket ok\n";
}
else {
    die $!;
}
# Give the socket an address.
if (bind(S, $this)) {
#    print "bind ok\n";
}
else {
    die $!;
}

# Call up the server.

if (connect(S,$that)) {
#    print "connect ok\n";
}
else {
```

```
    die $!;
}

# Set socket to be command buffered.

select(S); $| = 1; select(STDOUT);

#    print S "@PJL ECHO Hi $hostname! $ends";
#    print S "@PJL OPMSG DISPLAY=\"Job $whoami\" $ends";
#    print S $dpi300;

# Avoid deadlock by forking.

if($child = fork) {
    print S $dosCr;
    print S $TimesNewR;

    while (<IN>) {
        print S;
    }
    sleep 3;
    do dokill();
} else {
    while(<S>) {
        print;
    }
}

sub dokill {
    kill 9,$child if $child;
}
```

11.6 Running an *if* for remote printers

One oddity of lpd is that the *if* is not run for remote printers. If you find that you need to run an *if*, 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/lpd/filter-lj-5:
lj-5-remote:lp=/dev/null:sh:rm=printer.name.com:\
        :rp=raw:sd=/var/spool/lpd/lj-5-raw:
```

in 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.

11.7 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.

11.8 From an Apple.

Netatalk supports printing from Apple clients over EtherTalk. See the *Netatalk HOWTO Page* `<http://thehamptons.com/anders/netatalk/>` for more information.

11.9 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* `<ftp://ftp.gwdg.de/pub/linux/misc/ncpfs/>`.

12 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.

12.1 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 should(tm) allow 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.

That said, I've never tried this since I don't have a winprinter, so your mileage may vary. All the software involved should be available from the Ghostscript home page (see the links in the Ghostscript section of this document).

12.2 The pbm2ppa program

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* `<http://www.rpi.edu/~normat/technical/ppa/>`; 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.

12.3 The pbm2l7k program

Most of the cheap Lexmark inkjets use a proprietary language and are therefore Winprinters. However, Henryk Paluch has written a program which can print in black and white on a Lexmark 7000. Hopefully he'll be able to figure out color and expand support to other Lexmark inkjets. See *here* `<http://bimbo.fjfi.cvut.cz/~paluch/l7kdriver/>` for more info.

13 How to print to a fax machine.

13.1 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!).

13.2 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* `<http://www.tpc.int/>`.

14 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:

14.1 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.

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.

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.

14.2 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

A German company 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.

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* `<http://www.channel1.com/users/rodsmith/wpfonts.html>` 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.

LyX

LyX is a front-end to LaTeX which looks very promising. See the *LyX Homepage* <http://www.lyx.org/> 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.

Jeff Phillips <jeff@I_RATUS.org> uses Caldera's WordPerfect 7 for Linux (on Slackware, of all things) and says that it works well. It apparently includes built-in printer support, as one would expect. Caldera should have info on <http://www.caldera.com/>. You can also buy a newer version of Wordperfect directly from Corel's chosen Unix port company.

Other vendors should feel free to drop me a line with your offerings.

15 On-screen previewing of printable things.

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

15.1 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.

15.2 TeX dvi

TeX DeVice Independant 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.

15.3 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.

16 Serial printers under lpd

16.1 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 0; line = 0;
intr = ^C; quit = ^\; erase = ^?; kill = ^U; eof = ^D; eol = <undef>;
eol2 = <undef>; start = ^Q; stop = ^S; susp = ^Z; rprnt = ^R; werase = ^W;
lnext = ^V; min = 1; time = 0;
-parenb -parodd cs8 hupcl -cstopb cread -clocal -crtscts
-ignbrk -brkint -ignpar -parmrk -inpck -istrip -inlcr
-igncr -icrnl ixon -ixoff -iuclc -ixany -imaxbel
-opost -olcuc -ocrnl -onlcr -onocr -onlret -ofill -ofdel nl0 cr0 tab0
bs0 vt0 ff0
-isig -icanon -iexten -echo -echoe -echok -echonl -noflsh -xcase
-tostop -echoprt -echoctl -echoke
```

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

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`'.

16.2 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).

17 Credits

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.