The Linux Printing HOWTO

Grant Taylor, <gtaylor+pht@picante.com> v3.9, 25 July 1996

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 Intro

Since this 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 PHT*

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 SunSite (`sunsite.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 response to too many printing questions in comp.os.linux, and posted it. This predated the HOWTO project by a few months and was thus 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 in Linuxdoc-SGML.

1.2 Copyright

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://sunsite.unc.edu/mdw/HOWTO/Printing-Usage-HOWTO.html>` covers this, and a few other queue manipulation commands you should probably know.

3 Kernel printer devices

3.1 The lp device

The Linux kernel, assuming you have compiled in or loaded the lp device (the output of `cat /proc/devices` should include the device lp), 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. Try both ;)

One cannot run the plip and lp drivers at the same time on any given port. You can, however, have one or the other driver loaded at any given time either manually, or by kerneld with v2 (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 lp device's interrupt usage, polling rate, and other options.

When built in to some 1.3.x and v2 kernels, 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 in version 2 and late-model 1.3.x kernels, 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 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.

3.2 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://sunsite.unc.edu/mdw/HOWTO/Serial-HOWTO.html>`.

4 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...

5 How to set things up, basic

5.1 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 nauseum. 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
while(<STDIN>){chop $_; print "$_\r\n";};
```

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.

5.2 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.

6 How to obtain an appropriate magic filter

6.1 APS Filter

One of the nicest magic filter packages is aps filter, by Andreas Klemm. The Linux Software Map entry goes something like this:

```
Begin3
Title:          apsfilter
Version:        4.9.1
Entered-date:   Montag, 10. Juli 1995, 21:22:35 Uhr MET DST
Description:    magicfilter for lpd with auto filetype detection
Keywords:       lpd magicfilter aps apsfilter
Author:         andreas@knobel.GUN.de (Andreas Klemm)
Maintained-by:  sledge@hammer.oche.de (Thomas Bueschgens)
                andreas@knobel.GUN.de (Andreas Klemm)
Primary-site:   sunsite.unc.edu
                /pub/Linux/system/Printing/
                211KB aps-491.tgz
Original-site:  ftp-i2.informatik.rwth-aachen.de
                /pub/Linux/pak/APSfilter/aps-491.tgz
Platforms:      C-Compiler, gs Postscript emulator, pbmutils
Copying-policy: GPL
End
```

APS filter installs as an *if* filter for a print queue, and will translate from many common file types into your printer's command set. It understands, for example, text, PostScript, dvi, gif, and others.

6.2 Magic-filter

The magic-filter package, written by B.A.McCawley, works much the same way as does APSfilter, but since it is written in bash, it might be more easy to customize. The LSM entry:

```
Version: 0.4
Description: Bash scripts to simplify the task of configuring lpd to
   auto-detect file types using /etc/magic. (As advertised in the
   printing-HOWTO).
   This package uses existing filters so it only prints file types for
   which one already has filters (or compressed versions
   thereof). Alterative magic filter approches are described in the
   printing-HOWTO.
Keywords: lpd, /etc/magic, printing, filter
Author: B.A.McCauley@bham.ac.uk (Brian McCauley)
Maintained-by: B.A.McCauley@bham.ac.uk
Primary-site: tsx-11.mit.edu pub/linux/sources/usr.bin
              magic-filter-0.4.tar.gz  5709
Copying-policy: GNU GPL
```

6.3 The other Magicfilter

For maximum confusion, there appear to be two packages named just magic filter out there. (Or are they the same?)

```
Title:          magicfilter
Version:        1.1b
Entered-date:   04APR95
Description:    A customizable, extensible automatic printer filter.
                Lets you automatically detect and print just about any
                data type you can find a conversion utility for.  This
                filter is written in C and is controlled completely
                from an external printer configuration file.
                This version adds automagic creation of configuration
                files based on the installed software on your system,
                courtesy of GNU Autoconf.
                This version is a bug fix from 1.1/1.1a; filters for
                non-ASCII capable PostScript printers have been added.
Author:         H. Peter Anvin <Peter.Anvin@linux.org>
Primary-site:   sunsite.unc.edu
                53000 /pub/Linux/system/Printing/magicfilter-1.1b.tar.gz
Copying-policy: GPL
```

7 Serial printers under lpd

7.1 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/linux/termios.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`.

7.2 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).

8 Vendor Solutions

Thissectionis, bydefinition, incomplete.Feelfreetosendindetailsofyourfavouritedistribution.

8.1 RedHat 2.x

RedHat has a GUI printer administration tool 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.

8.2 Other Distributions

Please send me info on what other distributions do.

9 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.

9.1 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.

9.2 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.

9.3 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. 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 *SunSite* `<ftp://sunsite.unc.edu/pub/Linux/system/Printing/>`.

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

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.

9.5 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 `<ftp://linux01.gwdg.de/pub/ncpfs/>`. 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
/usr/local/bin/nprint -S net -U name -P passwd -q printq-q -
```

9.6 To an EtherTalk (Apple) printer

The netatalk package includes something like `nprint` and `smbclient`. Werner Eugster has documented the procedure for printing to and from an Apple network far better than I ever will; see his *web page* `<http://garnet.berkeley.edu/ weugster/appleprint.html>`.

Obscure caveat of the week: Netatalk does not work with SMC Etherpower PCI Card with a DEC tulip chip.

9.7 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:
```

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.

9.7.1 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) and stuff your print job into it. 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;
}
```

9.8 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:remote-hplj:\
        :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*.

10 How to print to a fax machine.

10.1 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.

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!).

10.2 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 *FAQ* `<http://www.town.hall.org/fax/faq.html>`.

11 How to generate something worth printing.

Here we get into a real rat's-nest of software. Basically, Linux can run four types of binaries with varying degrees of success: Linux, iBCS, Win16/Win32s (with dosemu and, someday, with Wine), and Mac/68k (with Executor). I'll just discuss native Linux and common Unix software, except to say that WordPerfect for SCO, and quite probably other commercial word processing software, runs fine under Linux's iBCS emulation.

For Linux itself, choices are mostly limited to those available for Unix in general:

11.1 Markup languages

Most markup languages are more suitable for large or repetative projects, where you want the computer to control the layout of the text to make things uniform. Trying to make a pretty sign in a markup language would probably hurt...

`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`.

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.

11.2 WYSIWYG

There is a general shortage of WYSIWYG text processing software. Feel free to write some, and do let me know of anything I've omitted here.

Lyrix

Lyrix is a front-end to LaTeX which looks very promising.

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.

Commercial offerings

At least Caldera and Red Hat ship packages containing the usual office apps like a WYSIWYGish word processor and a spreadsheet. I would assume they do a dandy job, but I've never used them. I think Caldera also ships Sun's WABI, so you could probably run something like MS Office under that if you had to integrate with other folks' files.

Jeff Phillips <`jeff@I_RATUS.org`> uses Caldera's WordPerfect 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/>`.

RedHat ships a suite called *Applixware*; you can find their web site at `<http://www.redhat.com/>`.

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

12 Ghostscript.

Ghostscript is an incredibly significant program for Linux printing. Most printing software under Unix generates PostScript, which is typically a $100-300 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 Adobe Acrobat's Portable Document Format, while 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).

Ghostscript's main failing is in its fonts, which are derived from bitmap scans of the characters. Superior, but commercially licensed fonts may be obtained from several places, since Ghostscript can read type 1 Adobe fonts. Adobe distributes useable fonts with their Acrobat PDF reader, or you may use the fonts from Adobe's ATM or a fontpack.

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.

12.1 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).

12.1.1 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.

12.1.2 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.

13 On-screen previewing of printable things.

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

13.1 PostScript

Ghostscript has an X11 driver best used under the management of the PostScript previewer Ghostview. The latest versions of these programs should be able to view PDF files, as well.

13.2 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.

14 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`>.