Chapter 13

Printers and Printing

Printing seems like one those constants of the computer world that should be so simple as to require next to no thought. Of course, you need to print things. Paper output is a part of doing business. Why, then, does printing continue to confound? Over the years, I have seen and read many different estimates regarding system administrators and printers. In particular, I'm interested in the question of how much time the average administrator devotes to this simple thing. One figure (which had a fairly strong argument backing it up) put printer administration at around 25 percent of a system administrator's time. Granted, this information goes back a few years, but I haven't seen much since then to convince me that this figure has changed.

This chapter examines how printing works and what processes are involved at the system level. I want you to walk away with an understanding, not of graphical printer admin tools, but of printing's dirty underbelly. On the subject of tools, I'll also explore some of the alternative tools that are available to make the whole printing experience just that much easier to deal with. The beauty of working with Linux is that you don't have to do things the way they are defined right out of the box. Many other options exist. Armed with the nuts and bolts of printers and printing, you should be able to print just about *any type of document* to just about *any type of printer.*

Selecting Printers for Linux (and a Note about "WinPrinters")

Beware.

Before you buy any printer to run with your Linux system, make sure you ask your dealer if it will work with Linux. I don't mean to frighten you into thinking that this is an impossible task, by the way. In fact, the "Does it work with Linux?" question is becoming less and less of a problem, but you can still save yourself an awful lot of headaches by asking first. Another way to protect yourself is to check out the hardware compatibility list (see Chapter 4) for Linux systems. This is a great idea for any kind of hardware add-on. For printers, though, you have an additional option that goes one step further.

Even seasoned Linux users like myself occasionally get "stuck" or "caught" with a printer that doesn't work. Some time ago, I bought an HP710C printer, and I just assumed (what do they tell us about assuming) that anything from HP would just work with Linux. After all, they make that great OS, HP-UX. As it turns out, being wrong is extremely easy. This particular printer used something called Printing Performance Architecture (PPA), a closed protocol whose secrets were only available to the Windows platform.

Looking around the Net, I found that an early driver had been put together by a guy named Tim Norman and a group of devoted developers. Unfortunately, at the time, you could only print in black and white, but Tim and his team were working on a color driver. So, I printed in black and white until a color driver finally became available. Had I known about Grant Taylor's printer compatibility list at LinuxPrinting.org (`http://www.linuxprinting.org/`), I wouldn't have had to wait because I wouldn't have bought the printer in the first place.

One of the most useful things at LinuxPrinting.org is a search engine that lets you select a particular manufacturer and get a report of all printers made by them, what level of support they offer, and what kind of tweaking you might need to do in order to make them work. This is especially good if you already have a particular printer and you want to know what filter or drivers are out there for your Linux system. (I'll delve into filters momentarily.)

If you are feeling less than adventurous (or you haven't already spent money on a printer), you can use the safer reporting option. Simply ask to see which color inkjet printers (or laser printers, or whatever type you are looking for) work perfectly. The resulting report lists printers by manufacturer with appropriate links to detailed descriptions of individual models.

The safest approach of all is to get a PostScript printer. These, unfortunately, don't tend to be inexpensive. I'll talk more about PostScript later.

How Printing Works

Linux offers a number of printing options. You can do text, PostScript, and local printers, as well as `lpd` remotes. If you want to, you can even create queues that direct printing to your coworkers' Windows 9*x* printers or provide Windows 9*x* users with Linux print services (using Samba). One printer is mostly okay, but add a handful and it gets a little bit more interesting. What Linux does is provide you with a wide range of options for dealing with this diversity.

Let's go back to the beginning and see how this whole thing actually works, starting with your old friend, the parallel printer.

Basic PC architectures usually have a single parallel port. A printer on that port would actually be connected to `/dev/lp0`. If your PC has more than one parallel connector (or if you added a printer card), the second and third ports would be `/dev/lp1` and `/dev/lp2`, respectively. Now let's pretend that you have a basic, run-of-the-mill text printer attached to the first parallel port. Printing can be as simple as this:

```
# echo "This is a test." > /dev/lp0
# echo "Ignore this print job." > /dev/lp0
# echo "This is nonsense." > /dev/lp0
# echo "^L" > /dev/lp0
```

Note: The last line means "Send a form feed to the printer to eject the page." The `"^L"` is actually a Ctrl-L. I inserted that into the text by pressing Ctrl-V (which allows me to then insert control characters) followed by Ctrl-L.

You may notice that something odd has happened. There is a distinct possibility that the output of this simple job looks something like this:

```
This is a test.
               Ignore this print job.
                                     This is nonsense.
```

This is what my printer configuration tool refers to as "stair stepping of text." You may also have heard it referred to as the "staircase effect." What happens is that my printer, a LaserJet 5L in this case, expects either PCL- (its native printer-command language) or DOS-style text. That means carriage returns and line feeds at the end of each line, unlike Linux files, which default to simply line feeds. This classic problem handily brings us to the topic of filters.

Filters

Let's take exactly the same lines of (non)information (minus the `echo "^L"` line) and create a file called "testfile." I used the Pico editor for this, but you can use whatever editor makes you happy. If you simply `cat` the file to the printer as before (with `cat testfile > /dev/lp0`), the results are the same. Building (or coding) a filter can be quite simple. Here's what mine looks like:

```
#!/bin/bash
echo -ne \\033\&k2G
cat
echo -ne \\f
```

The first echo line sends the command that tells my LaserJet to convert Linux line feeds to a carriage return followed by a line feed, an escape character followed by the printer control sequence &k2G. This is an HP-specific code, so you may have to check your printer manual for your printer's correct code sequence. The next line simply takes the input it got and passes it through unaltered. The final line sends a form feed to eject the page. I called the file dosfilter, moved it to /usr/local/bin, and made it executable.

```
# cp dosfilter /usr/local/bin
# chmod 755 /usr/local/bin/dosfilter
```

I'll resend the job and use my newly created filter.

```
# cat testfile | /usr/local/bin/dosfilter > /dev/lp0
```

Now my text comes out looking normal: one line after the other and properly aligned. Pretty neat, but in real life, you don't usually send jobs to printers in this way. You create printers on the system and send jobs by spooling them. Let's create a spooler definition called lptest. Do this by editing /etc/printcap and adding the following information. Keep in mind that the printcap file is pretty picky about how it is laid out. That's why the guide you received with your Linux system (assuming you bought a boxed set) tells you to use printtool or something like it. Learning to do it the hard way is more fun in the long run and you wind up really understanding how the whole thing works. Later in the chapter, I'll show you how to use nice tools.

The /etc/printcap file looks like this:

```
lptest:\
     :sd=/var/spool/lpd/lptest:\
     :mx#0:\
     :sh:\
     :lp=/dev/lp0:\
     :if=/usr/local/bin/dosfilter:
```

Here's what all this means. The first line is simply a printer name. If you send a job to a printer with the lpr command, you would specify this queue name. The next line refers to the spool directory. In other words, where all the information related to the current job goes. This is a directory under the /var/spool/lpd directory (by default), but it could conceivably go anywhere you like. /var/spool/lpd is convention. I like to use a directory name that is the same as the queue name. mx refers to the maximum file size that you will allow sent to the printer. A zero means "unlimited," which is generally what you want unless you have decided that specific printers will not take jobs beyond a certain size (too slow, perhaps). The next line (:sh:\) means "suppress header page." Because this is a physically connected device, you have the lp option, which defines the parallel device itself (/dev/lp0 because I only have one parallel port). The last line refers to the "input filter," the dosfilter created earlier.

To make all this work, all you really need to do is create the spool directory (/var/spool/lpd/lptest) and send your job to the printer.

```
# mkdir /var/spool/lpd/lptest
# lpr -Plptest testfile
```

Did you get an extra, unwanted form feed with your job? If so, modify your dosfilter and remove (or comment out) the form feed line. Another way to deal with this, if you do not want to modify your dosfilter, is to simply add another parameter to the /etc/printcap definition that tells the printing subsystem to "suppress form feed." The line should look like this:

```
:sf:\
```

The option sh and sf are Booleans. Their presence in the /etc/printcap file means "true." It's up to you to decide what options you will need. For a complete list of printcap options, you can use man and check out the page.

```
# man printcap
```

By default, when you send a job to the printer with the lpr command, it uses a queue definition called lp (as opposed to lptest, or whatever name you gave your queue). If the queue is simply named lp, the only command you need to print is this:

```
# lpr printfile_name
```

That's why you used the long form of the command.

```
# lpr -Pprint_queue_name printfile_name
```

Here's a nice tip that will save you a few keystrokes. If you are always printing to one specific printer, you can add the PRINTER environment variable to your .bash_profile.

```
PRINTER=lptest ; export PRINTER
```

Now, after you log in, all you have to do (assuming you want to print to lptest) is type the first, simpler version of the print command (lpr printfile_name).

Let's look at printing to a remote Linux (or UNIX) printer. In this example, I'll create a printer called "faraway" on the current machine, "nearlinux." The printer (the lptest created earlier) is on another machine called "farlinux." Here's the /etc/printcap entry for nearlinux:

```
faraway:\
  :sd=/var/spool/lpd/faraway:\
  :mx#0:\
  :sh:\
  :rm=farlinux:\
  :rp=lptest:
```

If this is the first time you are trying this, I should tell you it won't work right away. You see, the printing subsystem does have some security associated with it. You must be *allowed* to print to farlinux. This is done by editing the file `/etc/hosts.lpd` and adding the host names or the host IP addresses of the machines that are allowed to print. My `hosts.lpd` file has these entries:

```
192.168.22.2
192.168.22.3
nearlinux
```

It is probably a good idea to restart your `lpd` daemon after making this kind of change. One way to do this is to simply do a `ps ax | grep lpd` and kill the current `lpd` process. The second (this may vary slightly from system to system) is to use these commands:

```
# /etc/rc.d/init.d/lpd stop
# /etc/rc.d/init.d/lpd start
```

On some releases, you can wrap this up in one command by using "restart" instead of "stop" followed by "start." Now from my machine, I can send a really important file to my remote Linux machine for printing.

```
# /usr/games/fortune -l | lpr -Pfaraway
```

HP JetDirect Adapters

A fairly common remote printing situation you may find yourself working with is the HP JetDirect adapter. These are increasingly common in IT shops these days. You'll find both a stand-alone version (to which you can attach any printer) and a built-in version common with network-ready HP LaserJets. You will need to have the JetDirect adapter or printer set up with an IP address. You should refer to the accompanying documentation to do this. Most of the HP network-ready printers enable you to do this setup through the printer's control panel. HP also distributes software packages called JetAdmin and WebAdmin to do this over the network. Once you've got your printer configured, using a JetDirect-connected printer with your Linux system is very easy. As far as Linux is concerned, these are just remote print servers.

In an earlier example, you set up a printer locally referred to as "faraway" to access a remote queue called "lptest." Here's what it looked like:

```
faraway:\
   :sd=/var/spool/lpd/faraway:\
   :mx#0:\
   :sh:\
   :rm=farlinux:\
   :rp=lptest:
```

Keeping that example in mind, let's say that your adapter has an IP address of 192.168.1.225 and a host name of `hpjd1`. To create a queue on your Linux system, use this `printcap` entry:

```
jdqueue1:\
   :sd=/var/spool/lpd/jdqueue1:\
   :mx#0:\
   :sh:\
   :rm=hpjd1:\
   :rp=text:
```

For text-only printing, this is all there is to it. Notice that the remote printer's name is "text." JetDirect cards and adapters have two recognized printer names. The other is "raw." If your output is always PCL or PostScript, you should use "raw" as the remote printer name.

HP also sells three-port HP JetDirect adapters that enable you to connect three printers at one IP address (or one physical network jack). Using one of these is not much more complicated. These cards recognize six different printer names: for simple text output, they are `text1`, `text2`, and `text3`. For raw, or PostScript output, try `raw1`, `raw2`, and (you guessed it) `raw3`.

Printer Job Control

The master control program for printers is `lpc`, a small, interactive command-line program. In its simplest form, you type this:

```
# lpc
```

The system replies with a quiet little prompt.

```
lpc>
```

If you type `status` here, you will get the status of all printers configured on that machine. Here's what the output looks like on my system:

```
lpc> status
   lp:
        queuing is disabled
        printing is disabled
        no entries
        no daemon present
color:
        queuing is enabled
        printing is enabled
        no entries
        no daemon present
lptest:
        queuing is enabled
        printing is enabled
        no entries
        no daemon present
```

Let’s take an example in which my main printer, `lp`, is having problems and I don’t want the system to continue trying to print to it while I am trying to fix it. At the `lpc>` prompt, I type the following:

```
down lp "You should not have bothered with this printer."
```

What this will do is take down the printer and stop jobs from getting to it. Notice the message that follows `down lp`. This will print a message to anyone who queries the status of the print queue from his or her computer (I am assuming a remote user here, but the user could be local as well). So, not knowing what my system administrator has done, I still send a job to the printer. I then decide to see where in the queue my job sits, so I use `lpq` to find out. As you might have guessed, `lpq` reports on the status of queued jobs. You invoke it like this:

```
# lpq -Pprinter_name
```

Remember also that if you specify a `PRINTER` environment variable, you don’t have to specify the printer. In other words, typing this command means that I can just enter the command `lpq` and leave it at that.

```
PRINTER=lptest ; export PRINTER
```

Here’s what happens when I check the status of my job after my system administrator (okay, it’s really me) downs the printer:

```
mycomputer.salmar.com: waiting for queue to be enabled on scigate
Rank   Owner     Job  Files                             Total Size
1st    root      16   /etc/profile                        546 bytes

Warning: lp is down: "You should not have bothered with this printer"
Warning: lp queue is turned off
no entries
```

This tells me that I should consider using a different printer. The other alternative is to remove the job from the queue. This is done with the `lprm` command. To remove my orphaned job (number 16), I type the following:

```
# lprm -Plptest 16
```

Printing Anything to Any Printer

I'd like to talk a little bit more about filters. In my office, I have a small HP LaserJet 5L. It's the one I've been using for all the examples in this chapter. If you've spent any time whatsoever with Linux (or other UNICes/UNIXes), you know that most applications print using the PostScript format. Unfortunately, my printer doesn't print PostScript. Luckily, Linux is distributed with a little package called `ghostscript`.

I won't spend much time on `ghostscript` except to tell you that it is a powerful tool that also makes a great print filter. If I try to print a PostScript file to my printer, it comes out as strange text, which just happens to be a kind of code—code written in the PostScript language. Looking at the first ten lines of a PostScript file on my system, I see this:

```
# head contact.ps

%!PS-Adobe-PS
%%BoundingBox: 54 72 558 720
%%Creator: Mozilla (NetScape) HTML->PS
%%DocumentData: Clean7Bit
%%Orientation: Portrait
%%Pages: 1
%%PageOrder: Ascend
%%Title: Registrant Name Change Agreement
%%EndComments
%%BeginProlog
```

This is also what it looks like if I just send it to my printer without a filter of some kind that can interpret PostScript. Different Linux distributions offer different alternative filters, but all should have `ghostscript` in common.

Here's an example. I'll send my `contact.ps` file to the printer but pass it through a `ghostscript` filter beforehand. (Note that the following line wraps. It is just one line.)

```
   # cat contactm.ps | gs -q -dNOPAUSE -sDEVICE=ljet4 -r300
-sPAPERSIZE=letter -sOutputFile=- - | lpr
```

I know it looks like a lot to take in, but it's really pretty simple. The `-q` means that `ghostscript` should perform its work quietly. Normally, `ghostscript` would put out a lot of "this is what I am currently doing" information, not what I want for a print job. The `-dNOPAUSE` tells `ghostscript` to process all pages without pausing to ask for directions. The first `-s` flag that you see specifies the printer type. The `ljet4` definition covers a whole range of LaserJet printers that can do 600 dpi resolution.

This brings me to the `-r` flag, where I define a 300 dpi resolution. This Netscape Navigator generated page (remember that you can print to a file when using Netscape Navigator) doesn't need a 600 dpi resolution. `ghostscript` also enables me to specify `papersize`, important for those of us in North

America who hold firmly (if not wisely) to the 8½x11-inch letter-size format. Finally, I specify standard out as my output file. Notice the last hyphen in that line. It means that `ghostscript` is taking the input through its standard in. The last thing I do is fire it to the printer.

The great thing about `ghostscript` is its extensive printer support. Visit the `ghostscript` printer support page (`http://www.cs.wisc.edu/~ghost/`) if you want to see the latest and greatest list of support. When you visit the site, scroll down the list and click the "printer compatibility" link.

Armed with this, I could use essentially the same line to create an output filter for printing. Remember the dosfilter example earlier? There was only one real active line in the filter script (other than the staircase effect change) and that was a simple `cat`. That line would now be the `ghostscript` line from the previous code minus the `| lpr` at the end of it.

```
gs -q -dNOPAUSE -sDEVICE=ljet4 -r300 -sPAPERSIZE=letter -sOutputFile=- -
```

You can even use `ghostscript` as a desktop X viewer for PostScript files and documents by simply typing `gv` followed by the name of the file you want to see. `gv`, as you might have guessed, stands for "`ghostscript` viewer" (see Figure 13.1).

```
gv netscape_out.ps
```

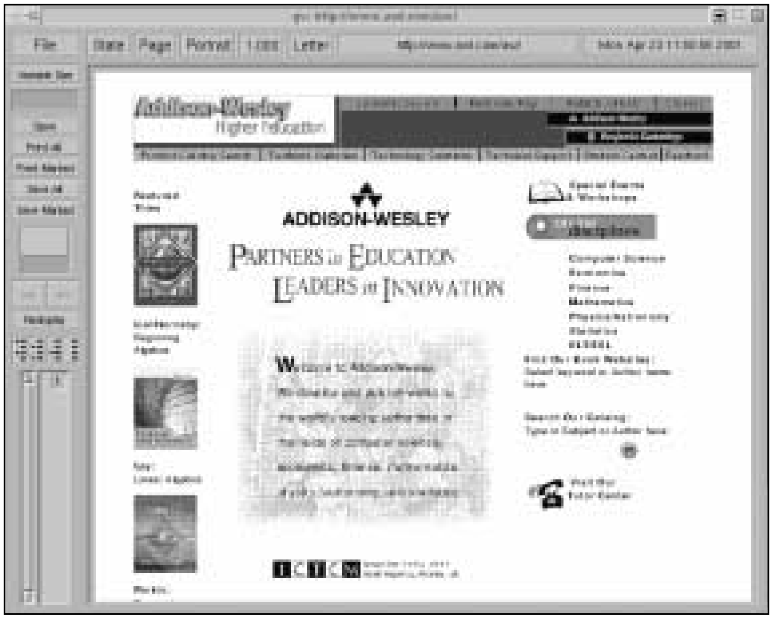


FIGURE 13.1 The `ghostscript` viewer, `gv`

Tying It Up: Advanced Filters with Ghostscript

All right, let's put some of the ideas presented in this chapter together. As I mentioned, a number of programs generate PostScript only rather than trying to create output for different printer types and languages. In the Linux world, that accounts for a lot of programs. Netscape Navigator is one such program, as is StarOffice from Sun Microsystems. So how do you deal with this?

Let's start with a quick reminder of the `ghostscript` output filter to convert PostScript to my LaserJet 5L printer. I've saved this file as `/usr/local/bin/psfilter` and made it executable with `chmod 755 /usr/local/bin/psfilter`.

```
#!/bin/bash
#
# Ghostscript filter so that my HP LJ5 will print PostScript files
#
echo -ne \\033\&k2G
gs -q -dNOPAUSE -sDEVICE=ljet4 -r300 -sPAPERSIZE=letter  -sOutputFile=- -
```

Now I also have a definition for the printer that uses that output filter.

```
pshpljet:\
:sd=/var/spool/lpd/pshpljet:\
:mx#0:\
:sh:\
:lp=/dev/lp0:\
:of=/usr/local/bin/psfilter:
```

Why PostScript?

The simple answer is that PostScript is close to a standard for defining objects (whether text or graphics) to printers, plotters, and even video screens. PostScript was designed to be completely device independent. Consequently, it provides a virtually universal print file format. PostScript (a trademark of Adobe Systems) was developed in the mid-80s and is actually a programming language. Because PostScript files are plain text, you can use a text editor to modify the source—even that of an image.

PostScript has been used in the UNIX world for many, many years. Because a number of the programs that come with your Linux system were originally designed in the UNIX world, it's not surprising that they work with PostScript. This is a plus and not a minus. As a result, you'll find many great tools specifically for working with PostScript. Furthermore, because you can print any PostScript file to any printer, everyone can do this.

A Few PostScript Tricks

Say you've got a two-page document (a program listing or a collection of bad, but funny, one-liners) and you really only want the thing to take up one page (they're not that good). That is where the `mpage` command comes into play. You are now going to print this document so that two pages fit onto one. The output will be rotated, so this will actually come out as portrait orientation.

```
mpage -2 funnyjokes.txt | lpr -Ppshpljet
```

The two pages appear on one page with a nice thin line around both pages. Let's make this just a little fancier, shall we? Try out the next command and see what kind of output it generates.

```
mpage -2 -B-5r-5l-3t-3b3 -M50l50r50t50b -H prog_specs.txt | lpr -Plptest
```

What does this all mean? The `-2`, by the way, could be a `-4` to fit four logical pages per physical page. Just so you are clear on my hastily chosen terminology, I am using the terms *logical* and *physical* to differentiate the printer output from the physical, "hold it in your hand" page. You can even do a `-8`, but that might be getting a tad silly. The `-B-5r-5l-3t-3b3` part means that a bold line (3 point) should be drawn around the box. That's actually the last 3 at the end of that line. The `-5r-5l-3t-3b` part means that you want a three-line margin at the top and bottom and a five-character margin at the left and right. The `-M` line uses a similar format and gives you a 50-point margin all around your virtual page. Remember that you have two logical pages on one physical page in this example. Finally, `-H` means that you want a header on each page. The header has the date on the right, the filename in the center, and a page number on the right.

So, where would you use this? One place that comes immediately to mind (after the one-liners, that is) is source code for your programs. This is a great way to generate compact program listings without having to import them into a word processor. I should point out that there is a non-PostScript way to create simple, numbered pages. All you have to do is use the `pr` command. Try this with a text-only printer definition:

```
pr +2 -h "Secret Kernel Enhancements" -o 5 ftl_travel.c | lpr -Ptextonly
```

`pr` is a command designed to format text for printing. The preceding line says to take the file `ftl_travel.c`, start printing at page 2 (`+2`), add a left-hand indent of 5 spaces (`-o 5`), and print the header "Secret Kernel Enhancements" on each page (`-h` followed by a string). If you do not specify a different header name, you'll get the filename itself, in this case `ftl_travel.c.` Oh, yes . . . it will actually start numbering the pages at page 2.

Alternative Print Systems

You'll get very little argument from the Linux community regarding the need for better and easier printing systems. While the `lp /lpd` printing system is good, works well, and has worked well for decades, a little simplification wouldn't hurt. Luckily, you can find some tools in the Linux world that are changing the face of printing from an administrative nightmare to something more fun and less frightening.

Every distribution differentiates itself with its own interface for system administration tasks such as setting up users and printers. Red Hat has `printtool` and SuSE has YAST. While I could explore each distribution's options for handling this, I'd like to spend some time looking at alternatives that are not release specific. In the next few sections, I'll introduce you to two such systems. Be sure to read Chapter 16 for an admin tools roundup and even more alternatives.

PDQ

According to the author Jacob A. Langford, PDQ stands for "print, don't queue" and a variety of other acronyms. His page is at the following address: `http://pdq.sourceforge.net/`.

PDQ is a nice, friendly little package that works on Linux and a variety of other UNIX systems. I'm not sure I agree completely with Jacob's reasoning on printing, accounting, and queuing, but that doesn't detract from PDQ's value. While it seems quite capable of dealing with large deployments of printers, what I particularly like about PDQ is that it presents the kind of friendly face that users of that other OS find so appealing, complete with a slick X window–like interface and wizards to help you set up your printers quickly and easily. You can even use Grant Taylor's compatibility list (at `http://www.linuxprinting.org/`) to find the latest PDQ drivers and filters.

Downloading the PDQ source and installing it is easy. When you get your copy, just make sure you substitute the appropriate release information. All you have to do is `untar` and unzip, and then do a `make` followed by a `make install`, as follows:

```
tar -xzvf pdq-2.2.1.tgz
cd pdq-2.1.2
make
make install
```

Now set up a base `printrc` configuration file, like this:

```
mv /etc/pdq/printrc.example /etc/pdq/printrc
```

To start PDQ and configure your first printer, simply type this command:

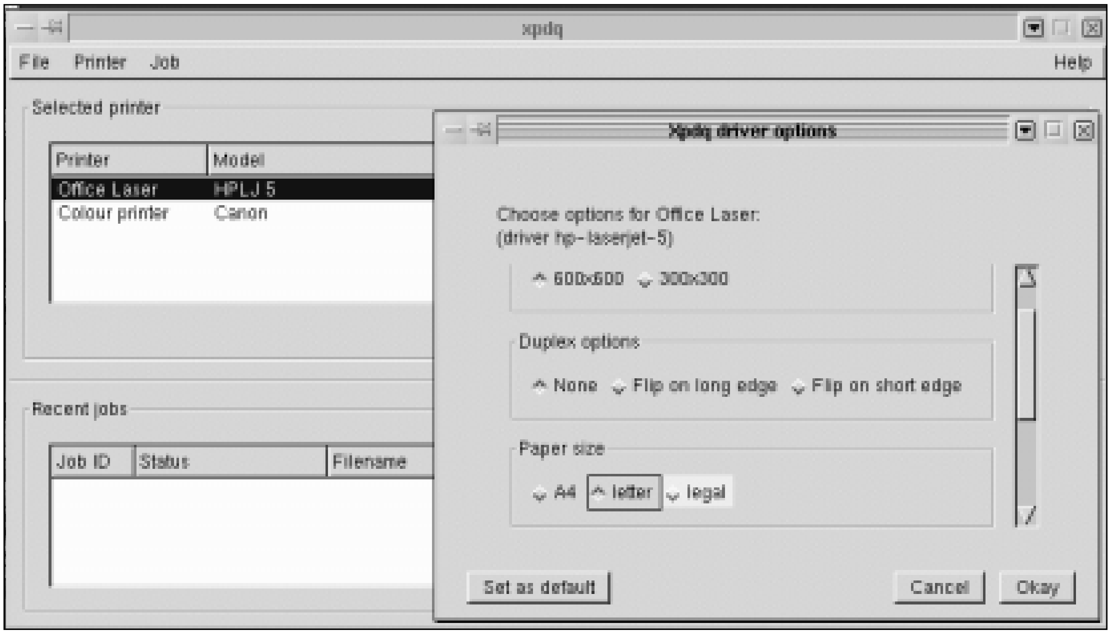
```
xpdq &
```

When the interface comes up, click Printer, choose Add, and follow the steps in the printer wizard. When you move your mouse over a field, PDQ provides context-sensitive, "bubble" help to guide you.

When you get to the driver selection screen, you may find the list somewhat limited—only about a dozen printers are listed, although you may be able to use a generic definition. For my HP LaserJet 5L, I went back to Grant Taylor's printer compatibility list and found my printer. To install this new driver, I typed the following:

```
cd /etc/pdq/drivers/hp
```

In the drivers directory, there are subdirectories for the printer classes (or brands, if you prefer). After changing to the hp directory, I used `vi` (you can use Emacs, Pico, or whatever editor you prefer) and simply cut and pasted the information on the screen. Then I restarted `xpdq`, and my printer was in the list (see Figure 13.2).



Printer	Model
Office Laser	HPLJ 5
Colour printer	Canon

Job ID	Status	Filename

Choose options for Office Laser:
(driver hp-laserjet-5)

[✓ 600x600] 300x300

Duplex options
[✓ None] Flip on long edge Flip on short edge

Paper size
A4 [✓ letter] legal

FIGURE 13.2 Printer configuration with xpdq

To print a job using PDQ, you send the job through the command line (did I mention there are command-line utilities that let you do what the X interface does?). The format is simple. For my test, I sent a PostScript file. I wanted to see PDQ's drivers and filters in action.

```
pdq -P hp5lj /tmp/oneliners.ps
```

The `-P` flag calls the printer name as I defined it in the wizard, while the file called `/tmp/oneliners.ps` is my very important print file. The X interface (`xpdq`) then reports the status of the job and allows me the opportunity to reprint it, get a status on it (if it is deep in the list of jobs), or get information on the type of job.

CUPS

Another alternative to good old-fashioned `lpd` printing is CUPS. CUPS, or the Common UNIX Printing System, is designed to be a platform-independent printing system that works across many different UNIXes (or UNICes, if you prefer). The company that produces CUPS (Easy Software Products) distributes it under the GPL, but you should be aware that the number of print drivers is limited. You'll find the latest software at this address: `http://www.cups.org/`.

For large printer support, you might want to consider their ESP Print Pro, a commercial offering that includes CUPS, lots and lots of printer drivers, and a nice GUI. If you want to explore the commercial side of CUPS, visit `http://www.easysw.com/printpro/`.

CUPS uses the Internet Printing Protocol (IPP), a next-generation printing system aimed at replacing LPD with a "universal" printing environment (although it still supports LPD), where any user anywhere can print to any printer anywhere. It also aims to provide better authentication and security. The proposed standard would even allow for encrypted print jobs. Those who are really curious as to where this is going can visit the Printer Working Group's Web site (`http://www.pwg.org/`).

As I mentioned, CUPS has limited printer support in its free package; however, some popular printers are supported. If you want to go this route but prefer to stick to the freeware version (and you may use the provided drivers), you can still get a nice GUI for your desktop. Actually, you can get *several* nice GUIs, which kind of gives you an idea of the growing popularity of CUPS.

Visit the CUPS-Related Stuff Home Page (`http://cups.sourceforge.net/`) and you will find something to enlighten your CUPS experience. In particular, check out KUPS and QtCups.

Miscellaneous Tips and Tricks

If you are running Red Hat 6.1, you may have found yourself at a loss in trying to figure out why your system does not want to recognize your printer port.

There are two potential problems at work here. The first has to do with a line missing from the `/etc/conf.modules` file. Try adding this line if you don't see it:

```
alias parport_lowlevel parport_pc
```

Another problem has to do with the version of `modutils` that was distributed with the system. Visit the Red Hat site and pick up the latest `modutils` RPM package.

Resources

CUPS

`http://www.cups.org/`

Ghostscript

`http://www.cs.wisc.edu/~ghost/`

Linux Printing HOWTO

`http://www.linuxdoc.org/HOWTO/Printing-HOWTO/index.html`

Linux Printing Usage HOWTO

`http://www.linuxdoc.org/HOWTO/Printing-Usage-HOWTO.html`

LinuxPrinting.org

`http://www.linuxprinting.org/`

PDQ

`http://pdq.sourceforge.net/`

Printer Working Group

`http://www.pwg.org/`