Printing from Unix

Chris Bidmead masters setting up multiple printers and takes a detailed look at an old master.

Last month I mentioned an email from Laurence Crummay (crums@greengates.karoo.co.uk), who wanted to know about setting up a printer on a Unix machine and sharing it with a Windows machine using Samba. I suggested O'Reilly's excellent *Using Samba,* which is also accessible online at www.ora.com/catalog/samba/chapter/book.

In fact, many of you are asking how to set up printers under Unix, and I realise I've rather neglected this side of things. Getting the printer to work under Samba, once it's operating properly on the Unix system, is the easy part of the job, as we'll see. Laurence already had one working printer hooked into the parallel port of his Red Hat 5.2 Linux server, and wanted to know how to go about adding an extra enhanced parallel port (EPP) card and attaching a second printer to it.

The advice I gave him was: 'This should turn out to be a fairly trivial exercise. The physical port that the new EPP card assigns is a low-level, PC (ie, non-Unix) interrupts/io_address issue, but I guess it will turn out to be one of |/dev/lp0|, |/dev/lp1| or |/dev/lp2|. If your existing parallel port device presents itself to Unix as |/dev/lp1|, then its I/O address will be 0x378, and I'd expect the new port to be |/dev/lp2| at 0x278.'

As Laurence already had a working printer, I suggested he take a look at /etc/printcap, which is the config file defining the Unix printer setup. It's important to point out that /etc/printcap doesn't define the physical port – it simply tells the printer daemon – lpd – where it is and what to do with it.

Well, if you're new to this perhaps I should start at the beginning. Several printing systems are available for Unix, but the traditional standard is the Berkeley System Distribution Line Printer (BSD lp), and I'll stick with that one for now as it's the one you're most likely to come across. There are several core elements to this (executables, config files, devices) and some optional fancy front ends, but the system boils down to the following areas.

A print spool directory

This is an out-of-the-way location (usually under somewhere such as /var/spool/lpd) where the system will chuck stuff that needs to get printed. Effectively this directory is the printer, if the rest of the print system is working properly. There can be several print spool directories, each behaving like a logical printer. Several logical printers can be attached to the same physical printer if you want to use its different characteristics. For example, the Lexmark Optacolor 45 I'm running here also doubles as a mono printer, and I've given it a separate print spool for each of these two functions. This would be an easy way to print a colour file in black and white. Perhaps a more appropriate use for multiple queues into the same device would be with one of those multi-purpose machines that can either print documents on the spot, or send them out as faxes.

The most important print tool is lpr, which is how an ordinary user passes information

lpd

This is an ever-vigilant daemon that handles stuff turning up in the print spool directories. The daemon is usually started at boot time, and is supposed to run constantly. The BSD lpd has a reputation for mysteriously vanishing from time to time, so if your system's ability to print suddenly evaporates, the first thing to do is to check that the lpd process is running. You can do this from the command line using a term such as:

```
ps ax | grep lpd
```

I'll break this down for you: ps is the command for showing the running processes; the ax switch (which does vary from one Unix to another) gives you a list of all the processes in the system; and piping the output through grep via the '|' character filters this list down to lines containing the string 'lpd'.

By the way, a stupid mistake you can make here – and I know, because I've done it – is to assume lpd is running simply because you get a result from the above pipeline. You might, for example, currently be checking the man page for lpd from another window or terminal, in which case the output line man lpd will be passed through grep. And obviously (well, it's obvious now, but this is the one that fooled me) the grep lpd process itself will show up in its own output.

The /etc/ printcap file

Printcap stands for 'printer capability' and is a textfile database telling you which printers are available, how they can be used, and what print spool directories they are associated with. Seasoned Unixen write these by hand, but utilities such as Red Hat's printtool can construct a working /etc/printcap file for you as you respond to its dialog boxes. It also automatically creates the relevant spool directories for each printer in /etc/printcap.

If you do use printtool, I urge you to take a look at the resulting /etc/printcap so that you at least get some idea of what's going on. Your system should provide a printcap man page to explain how it works.

Print tools and utilities

These are used for checking and controlling the print system. The most important is lpr, which is how the ordinary user passes material into the root-privileged print spool directory, using a command such as lpr -Plp2 printable.stuff.

Here 'printable.stuff' is a file to be printed, and the parameter to the -P switch directs lpr to send it to the printer called lp2. You can omit this switch, in which case the output gets directed to the default printer, which is either the one you've set up in /etc/printcap as just plain lp, or another printcap device whose name you've stuffed into the PRINTER environmental variable with a command such as:

```
    export PRINTER=lp3
```

This PRINTER variable is then checked by lpr, which collects up other information from the command line, creating a control file that also gets sent to the print spool. Lastly, lpr sends a message to the lpd daemon to say that the print spool is ready to be serviced.

Even if you're not using lpr directly (printing from GIMP, for example), this utility and the rest of the printing subsystem are still being evoked behind the scenes. Therefore, it's as well to know how to check from the command line that they're all working.

There are some other print utilities included in the BSD lp distribution. Line printer control program (lpc) is a general tool that can check the status of the lpd and the print queues, as well as turning printing on and off. To remove jobs from print queues, there's the lprm utility. lpq is a print queue inspector, useful for investigating what's going on if nothing is actually coming out of the printer.

Laurence got back to me to tell me of a successful result with his new Epson printer: 'Many thanks for your help. I edited /etc/printcap as follows:

```
    :epson:\
    :sd=/var/spool/lpd/epson:\
    :mx#0:\
    :sh:\
    :lp=/dev/lp2:\
    :if=/var/spool/lpd/✓
    olivetti/filter:
```

(*Key:* ✓ *code string continues*)

I then created the various directories in /var/spool/lpd, and hey presto! It works great now.

'As you suggested, my existing parallel port was /dev/lp1 – and the new one is /dev/lp2. The I/O addresses you suggested were also correct. As for the Samba side of things, it just happened! I think the entries in /etc/printcap become the printer shares for Samba [yes, they do]. They appeared in Network Neighborhood and setup fine.'

The low-down on USB

Peter Gray (peterg@paston.co.uk) writes: 'I have been reading all the articles on Unix (Linux) that I can find, but so far I haven't found any reference to support for USB. My current machine (that I assembled myself) uses USB for mouse, keyboard and monitor, and I would like to add Linux to it, but not at the cost of having to reconfigure it every time I change OS.'

Peter, USB support in Linux is still in its early stages. But you should be able to find the information that you need at www.linux-usb.org. Please get back to me and let me know how you get on.

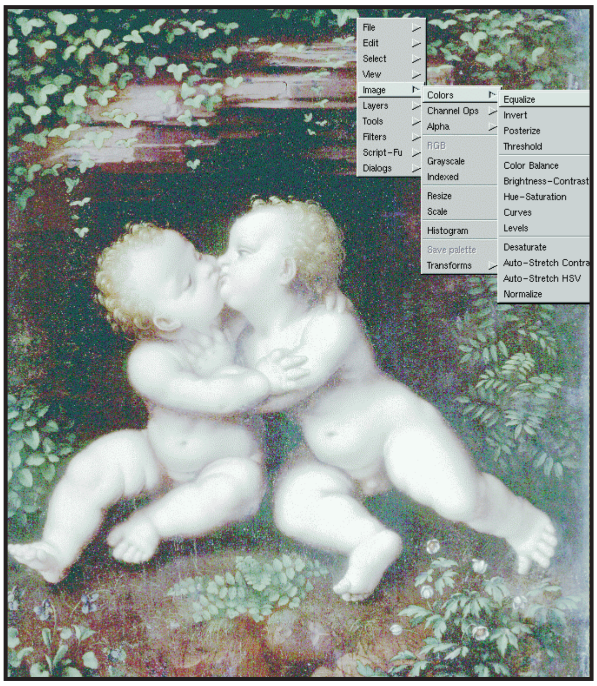
Bring out GIMP

A familiar cry for help comes from Roger King (kingfamily@clara.co.uk), who writes: 'Hi Chris. After the fourth attempt I have got Linux and Xwindows working. Finally, now that I've got it installed, WHAT DO I DO WITH IT?'

It's a question I understand, as it doesn't seem so long ago that I was in the same position. These days I use Unix for everything I do on computers, but that probably doesn't help answer the question. So here's a (very) specific case study. My mate Marcus owns an old painting attributed to The School Of Leonardo. It's worth a few bob as it stands, but would go up into telephone numbers if, as Marcus believes, Mr Da Vinci himself actually had a hand in painting it. So he's put together a website (at www.lostleonardo.com) that gathers together scholarly opinions about the painting and its period.

He's got a picture of the painting up there too, of course. But there was a fashion a few hundred years ago for coating old masters in dark varnish, and the detail isn't very clear. What Marcus needed was a sort of X-ray. He had a huge, 75MB scanned TIFF of the picture – was there some way that we could bring out the detail using a computer?

At this point we needed to bring out the GIMP. This is a sensational piece of free software you can download from www.gimp.org – the name stands for the GNU Image Manipulation Program. Loading this enormous file into the GIMP was tediously slow – but the problem wasn't Linux, but the relatively modest 200MHz Pentium Pro and 64MB of RAM it was running in. It would have been nice to try this on the speedy dual-processor Fujitsu-Siemens Celsius workstation I had here last year, but alas that's had to go back to the manufacturer.



The Gimp loads the original old master – all 75MB of it!

However, I realised that it wasn't necessary to work on a file this size, because it had a much higher resolution than anyone would be able to see in a browser (Marcus used high resolution to show individual sections of the picture on the website). So I used Gimp to create a reduced version less than 1MB in size.

The first thing we did, now that we had a manageable file, was equalise the colours (Image/Colors/Equalise from the right mouse button menu). This gave us our X-ray – a false colour rendition that brought up all the detail we were missing.

Next month, I'll tell you how we used the GIMP's multi-layer feature to create a version that brings out most of the detail – without departing too much from what were probably the original colours.

CONTACTS

Chris Bidmead welcomes your comments on the Unix column. Contact him via the *PCW* editorial office or email: **unix@pcw.co.uk**